

(19)



Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number:

0 388 560
A2

(12)

EUROPEAN PATENT APPLICATION

(21) Application number: 89313390.0

(51) Int. Cl.⁵: **G07G 1/12, G06K 7/10,**
G01G 19/413

(22) Date of filing: 20.12.89

(30) Priority: 24.03.89 US 328177

(43) Date of publication of application:
26.09.90 Bulletin 90/39

(84) Designated Contracting States:
BE CH DE FR GB IT LI LU NL SE

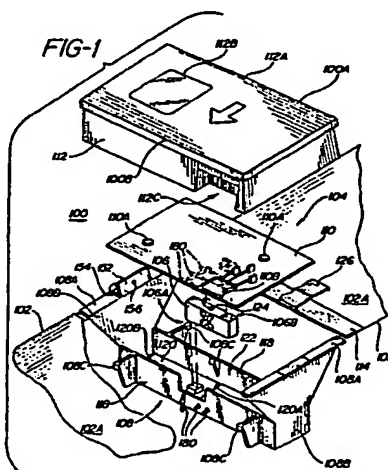
(71) Applicant: **SPECTRA PHYSICS INC.**
3333 North First Street
San Jose California 95134-1995(US)

(72) Inventor: **Taussig, Andrew Peter**
2515 West 22nd Avenue
Eugene Oregon 97405(US)
Inventor: **Isaacs, Blake L.**
748 Granite Place
Springfield Oregon 97477(US)

(74) Representative: **Murgatroyd, Susan Elizabeth**
et al
Baron & Warren 18 South End Kensington
London W8 5BU(GB)

(54) Data gathering system interface.

(57) A data gathering system (100) for use in a checkout counter (102) to determine information relating to products to be purchased and to provide such information to a cash register system includes a scale (104) supported within the checkout counter (102) for determining weights of products presented to the data gathering system (100). The scale (104) includes a subplatter (110) located below the upper surface of the checkout counter (102). An optical scanning arrangement (112) is supported on the subplatter (110) for reading coded labels on the products. A common interface circuit (200) is responsive to both the scale (104) and the optical scanning arrangement (112) for providing weight data and coded label data to the cash register system.



Xerox Copy Centre

EP 0 388 560 A2

BEST AVAILABLE COPY

DATA GATHERING SYSTEM INTERFACE

The present invention relates generally to data gathering systems for use at point of sale checkout counters and, more particularly, to a data gathering system which comprises an optical scanner supported on a weighing scale, and support means for suspending the system within a counter such that the optical scanner provides a weighing platter for the scale at an upper surface of the counter. Thus, bar coded data
 5 imprinted upon product labels presented at the counter can be read by the optical scanner, and the weight of such products can be determined by placing the products on the optical scanner. In effect, the optical scanner becomes the weighing platter of the scale. The bar code data and the weight data are supplied to the cash register system via a common interface.

Supermarket checkout counters commonly include optical scanners mounted therein for optically
 10 scanning bar code labels on products to be purchased as the products are moved over scanning windows in the top surfaces of the scanners. The scanners read and convert the bar code labels into product identification and pricing information which is used for the sale and other merchandising purposes, such as inventory control. Since many products sold in supermarkets, for example produce, are sold by weight, oftentimes weighing scales are also positioned adjacent the checkout counters. The weighing scales can be
 15 manually operable such that sales clerks weigh products and then enter the resulting prices on associated cash registers. Alternately, the weighing scales can be partially or fully automated such that the weights and resulting price information are directly passed to the registers.

One example of an automated data gathering system is disclosed in U. S. Patent No. 4,656,344, wherein a scale and an optical scanner are not only directly interconnected to an associated register but
 20 also are integrated into a single unit which fits within the checkout counter. While this integrated scale/scanner system offers advantages over the prior separated component systems in terms of convenience and space savings, still there are disadvantages and inconveniences associated with the integrated system. For example, by making the scale and scanner a single integrated unit, the system is heavy and difficult to install and/or remove from a receiving checkout counter. Adding to this problem, the system must
 25 be removed from the checkout counter each time the scale is to be zeroed or calibrated and then the system must be reinstalled.

The need for a data gathering system for use at a point of sale checkout counter which provides for both scanning coded labels and also weighing products which are to be sold by weight has been recognized, and a combined scanner and scale which meets this need is the subject of the above identified,
 30 commonly assigned, U.S. Patent Application entitled Scale Having Platter Including Removable Optical Scanner (Docket SPR 029 PA). The combined scanner and scale accomplishes is provided in a compact packaging arrangement which eases and facilitates removal and installation of the system for maintenance and repair.

It will be appreciated, however, that since the scanner and scale both communicate with the cash
 35 register system, such as for example an IBM Model 4683 System, it is necessary to provide for a communications interface between both of these components and the cash register system, even though the components are physically combined. Not only has this required redundant interface circuitry, but the use of redundant cabling, and interface ports in the cash register system. It is seen, therefore, that there is a need for an interface arrangement which avoids the difficulties heretofore encountered.

This need is met by a data gathering system according to the present invention for use in a checkout
 40 counter to determine information relating to products to be purchased and to provide such information to a cash register system. The data gathering system includes a scale means supported within the checkout counter for determining weights of products presented to the data gathering system. The scale means includes a scale weighing platter located below the upper surface of the checkout counter, optical scanning
 45 means supported upon the scale weighing platter for reading coded labels on the products, and a common interface circuit, responsive to both the scale means and the optical scanning means, for providing weight data and coded label data to the cash register system.

The scale weighing platter includes scanner locator means for positioning the optical scanning means
 50 on the scale weighing platter for assembly of the data gathering system. A support means is provided for suspending the data gathering system within the checkout counter. The scale means is secured to the support means.

The optical scanning means includes a bar code decoder circuit for decoding scan signals to provide coded label data, a scanner microprocessor for correlating coded label data and supplying the coded label data to the common interface circuit, and scanner memory means for storing control software for use by the scanner microprocessor.

The scale means may supply weight data to the bar code decoder circuit of the optical scanning means, and the bar code decoder circuit then supplies the weight data to the common interface circuit via the scanner microprocessor without alteration. Alternatively, the scale means may supply weight data directly to the common interface circuit.

5 The common interface circuit comprises an interface microprocessor, responsive to coded label data from the optical scanning means and to weight data from the scale means, interface memory means for storing control software for use by the interface microprocessor, and a driver circuit, responsive to the interface microprocessor, for supplying weight data and coded label data to the cash register system.

10 The data gathering system according to the present invention for use in a checkout counter determines information relating to products to be purchased and provides such information to a cash register system, including weight data and coded label data. The counter defines an upper surface upon which products are placed for access to the data gathering system. The data gathering system includes support means for suspending the data gathering system within the checkout counter, and scale means secured to the support means for determining weights of products presented to the data gathering system. The scale means 15 includes a scale weighing platter located below the upper surface of the checkout counter. An optical scanning means is supported upon the scale weighing platter for reading coded labels on the products. The optical scanning means has an upper surface including an optical scanning window, and is sized such that its upper surface is substantially aligned with the upper surface of the checkout counter when supported upon the scale weighing platter. The data gathering system includes a common interface circuit, responsive 20 to both the scale means and the optical scanning means, for providing both weight data and coded label data to the cash register system.

The scale weighing platter may include scanner locator means for positioning the optical scanning means on the scale weighing platter for assembly of the data gathering system.

25 The optical scanning means includes a bar code decoder circuit for decoding scan signals to provide coded label data, a scanner microprocessor for correlating coded label data and supplying the coded label data to the common interface circuit, and memory means for storing control software for use by the scanner microprocessor.

30 The scale means supplies weight data to the bar code decoder circuit of the optical scanning means, and the bar code decoder circuit supplies the weight data to the common interface circuit via the scanner microprocessor without alteration. Alternatively, the scale means supplies weight data directly to the common interface circuit.

35 The common interface circuit comprises an interface microprocessor, responsive to coded label data from the optical scanning means and to weight data from the scale means, memory means for storing control software for use by the interface microprocessor, and a driver circuit, responsive to the interface microprocessor, for supplying weight data and coded label data to the cash register system.

The data gathering system may further include cables connected between the scale means and the optical scanning means for conducting electrical signals and power, the cables being sized, positioned and secured to prevent interference with the operation of the scale means.

40 Accordingly, it is an object of the present invention to provide an improved combined scale and scanner system in which communication with a cash register system is facilitated; to provide such a combined scale and scanner system in which a single interface communicates between the scale and scanner, and the cash register system; and to provide such a combined scale and scanner system in which the interface is microprocessor controlled.

45 In order that the invention may be more readily understood, it will now be described by way of example with reference to the accompanying drawings, in which:

Fig. 1 is an exploded perspective view of a data gathering system in accordance with the present invention for use in a checkout counter;

Figs. 2-4 are top, side and end views, respectively, of the data gathering system of Fig. 1; and

50 Fig. 5 is an electrical schematic representation of the common interface circuit, and associated circuitry, which is responsive to both the scale means and said optical scanning means, and which provides weight data and coded label data to the cash register system.

Reference is now made to the drawings which show a data gathering system 100 in accordance with the present invention. The system 100 is designed for use in a point of sale checkout counter 102 and fits entirely within the counter 102. The system 100 is structured as two separate units which can be 55 independently manufactured, packaged and shipped and also individually handled and installed. By structuring the system as two separate units, it initially can be easily installed in the counter 102 and thereafter easily removed and reinstalled for system maintenance and repair. Further, the scanning operation can be more easily and accurately performed since scanning light beams pass through a single

scanning window as opposed to two or more windows and/or apertures in prior art systems.

The close proximity of the two separate units facilitates the transmitting both weight data and coded label data to a cash register system, such as for example an IBM Model 4683 cash register system. The data gathering system 100 conveniently provides for both reading bar-coded labels secured to products to be purchased and also weighing products which are placed upon the upper surface of the system.

This data is then sent to the cash register system via a common interface circuit, as will be explained more fully below. Not only does this utilize the system circuitry in a most efficient manner, but also reduces the amount of coaxial cabling and the number of cash register system input/output ports required. The data gathering system 100 comprises scale means, taking the form of a load cell scale 104 in the illustrated embodiment, which is supported within the checkout counter 102 by support means and provides for determining weights of products presented to the data gathering system 100. The scale 104 comprises a cantilever beam load cell 106 secured at one end 106A to the support means which comprises a support cradle 108 in the illustrated embodiment, and to a scale load receiving plate or subplatter 110 at its opposite end 106B. The subplatter 110 is located below the upper surface 102A of the checkout counter 102 as best shown in Figs. 3 and 4.

Optical scanning means comprising a self-contained optical scanner 112 is supported upon the scale subplatter 110 for reading coded labels, such as bar-coded labels, on products presented for purchase at the checkout counter 102. The optical scanner 112 has an upper surface 112A including an optical scanning window 112B through which scanning light beams pass and is sized such that its upper surface 112A is substantially aligned with the upper surface 102A of the checkout counter 102 when the optical scanner 112 is supported upon the subplatter 110. The weight of the optical scanner 112 is subtracted from the weight which is measured by the scale 104 or treated as a tare weight for the scale 104 such that the upper surface 112A of the optical scanner 112 serves as an extension of the subplatter 110 for receiving products to be weighed.

The subplatter 110 includes scanner locator means comprising two raised circular bosses 110A which are received by corresponding indentations (not shown) formed into the bottom of the optical scanner 112 for positioning the optical scanner 112 on the subplatter 110 for assembly of the data gathering system 100. Of course, differently formed bosses or different locating means can be provided as will be apparent to those skilled in the art.

Preferably, the data gathering system 100 is positioned within the checkout counter 102 such that the upper surface 112A of the optical scanner 112 is slightly above the upper surface 102A of the counter 102. The tapered trim strip 114 is secured across the checkout counter 102 adjacent the entry side 100A of the data gathering system 100 to elevate slightly the upper surface 102A of the counter 102 above the upper surface 112A of the system.

The support means or support cradle 108 is adapted to be hung from the checkout counter 102 by means of support flanges 108A which extend from end plates 108B of the support cradle 108. It should be apparent that the counter 102 can be adapted to support the data gathering system 100 from the support flanges 108A such that the system can be precisely located relative to the counter 102 with convenient adjustment, if necessary, being provided by shims or otherwise. The support cradle 108 comprises at least two subplatter stop members 108C, six stop members 108C being included in the illustrated embodiment as best shown in Figs. 1 and 2, positioned to engage the subplatter 110 at the maximum extent of its travel to thereby prevent potentially damaging overtravel of the load cell scale 104. Preferably, stop pads 108D made of rubber or other resilient material are secured to the upper surfaces of the stop members 108C as shown in Figs. 2-4.

The support cradle 108 comprises generally vertical side walls 118 and a bottom wall 120 which define a channel 122 extending laterally across the checkout counter 102 for receiving and protecting the load cell flexure 106 and electrical circuitry (not shown) which is connected to and operable with one or more load cells 124, see Fig. 1, secured to the load cell flexure 106 in accordance with well known weighing scale technology. The bottom wall 120 is peaked near its center 120A such that it gradually tapers downward toward the end plates 108B of the support cradle 108 adjacent which the bottom wall 120 terminates in open slots 120B. A crowned channel cover 126, shown in Figs. 1-4, includes an opening 126A through which the load cell flexure 106 is connected to the subplatter 110, see Fig. 2.

This support arrangement or mounting for the data gathering system 100 is preferred since it provides improved spill control over the prior art. In particular, any spilled liquids which flow over the entry side 100A or exit side 100B of the system will flow harmlessly down the sides of the optical scanner 112 to the floor beneath the system where it can be periodically or immediately attended to through access panels (not shown) in the counter 102. Spilled liquids which flow down the ends of the system will be limited due to the narrowness of slots 128 between the optical scanner 112 and the support flanges 108A/end plates 108B of

the support cradle 108, see Figs. 2 and 3. Further, the majority of such liquid will also flow harmlessly to the floor beneath the system due to the narrowness of the width of the channel 122 which is approximately one third of the width of the data gathering system 100. The remaining small portion of spilled liquid which passes through the narrow slots 128 will initially engage the crowned channel cover 126 and be diverted to the sides of the channel 122 and once again to the floor beneath the system.

Any spilled liquid which does manage to seep past the channel cover 126 will flow down the interior surfaces of the end plates 108B and/or be diverted by the tapered bottom wall 120 to pass to the floor beneath the system through the slots 120B. Spilled liquids are thus eliminated from the data gathering system 100 by paths which do not tend to interfere with the movement and hence the operation of the scale 104 of the system.

To maintain the accuracy of the scale 104, cables for conducting electrical signals and power between the optical scanner 112 and the scale 104 of the data gathering system 100 are formed and secured to the scale 104 during its manufacture. More particularly, cables 180 are sized such that they extend between and are secured to the subplatter 110 and one of the side walls 118 of the channel 122 such that the cables permit free movement of the scale flexure 106 but do not affect such movement, see Figs. 1-3. By thus sizing the cables 180 such that they do not rest upon a portion of the data gathering system 100 dependent upon the deflection of the load cell flexure 106, the weight of the cables 180 can be compensated by calibration of the scale 104. This cabling arrangement is important since cables to the optical scanner 112 must be routed through the subplatter 110 via an opening 110B therethrough. The cables 180 pass through a corresponding opening 112C, see Fig. 1, in the bottom of the optical scanner 112 and are precisely located and secured therein to assist in stabilizing the accuracy of the scale 104.

Reference is made to Fig. 5 which illustrates schematically, the common interface circuit 200 of the present invention and circuitry associated therewith. The common interface circuit 200 is responsive to both the scale means 104 and the optical scanning means 112 including scanner circuit 202, for providing weight data and coded label data to the cash register system 204, which may be an IBM Model 4683 cash register system, as stated previously. The optical scanning means includes circuit 202 having a bar code decoder circuit 206 for decoding scan signals to provide coded label data. Circuit 202, preferably an NCR VLSI decoder circuit, decodes label segment data received on line 208. Scanner microprocessor 210 correlates the coded label data received from circuit 206 and supplies the coded label data to the common interface circuit 200 under control of control software stored in scanner memory means 212, preferably comprising an EPROM. Scanner microprocessor 210 is preferably an INTEL 8039 microprocessor.

The scale means 104 may supply weight data via line 214 to the bar code decoder circuit 206 of the optical scanning means, and the bar code decoder circuit 206 then supplies this weight data to the common interface circuit 200 via the scanner microprocessor 210 without alteration. As indicated in Fig. 5 by line 214', the scale means 104 may supply weight data directly to the common interface circuit, if desired.

The common interface circuit 200 comprises an interface microprocessor 216, responsive to coded label data from the optical scanning means 112 and to weight data from the scale means 104. Microprocessor 216 is preferably a ZILOG Super8 microprocessor. Circuit 200 also includes an interface memory means 218, preferably an EPROM, for storing control software for use by the interface microprocessor 216. A driver circuit 219 is responsive to the interface microprocessor 216 for supplying weight data and coded label data to the cash register system 204.

The control software stored in scanner memory means 212 may be exemplified by the following listing.

45

50

55

IBM 4683 INTERFACE SOFTWARE FOR SUPER8

5

2500 A.D. Super 8 Macro Assembler - Version 4.01c

10

Input Filename : s84683.asm
Output Filename : s84683.obj

```

15      1      ; Super 8 Software for the IBM 4014 interface. The .hex file produced
2      2      ; is called S84683.HEX.
3
4      3      ; Spectra-Physics Part Number R96-0159 Rev A
5      4      ; Compiled by Blake Isaacs
6      5      ; Date: 3-14-1988
7
20     6      ; Interrupt vectors:
7
8
9
10     0000      dummy    equ    0
11
12     0000 0450 045F 0461      dw    vec_0,vec_2,vec_4,vec_6
13     0006 0463
25     13     0008 0465 0462 0467      dw    vec_8,int_p33,vec_c,vec_e
14     000E 0469
15     14     0010 0468 0460 046F      dw    vec_10,vec_12,vec_14,vec_16
16     0016 0471
30     15     0018 0473 0475 0477      dw    vec_18,vec_1a,vec_1c,vec_1e
17     001E 0479
16
17     0020      start:
18     0020 9F      ei          ; Zilog suggestion
19     0021 8F      di
20
35     21      ; Initialize PORT registers:
22     22      ; Initialize ports before setting to outputs:
23
24     0022 B0 D0      clr    P0
25     0024 B0 D2      clr    P2
26     0026 B0 D3      clr    P3
40     27     0028 B0 D4      clr    P4
28
29     002A E6 F8 88      ld     P2AH,#288      ; P31 is output (Tx0)
30     002D E6 F9 62      ld     P2BH,#262
31     0030 E6 FA 09      ld     P2CH,#209
45     32     0033 E6 FB 80      ld     P2DH,#280
33
34     ;      88 P2AH 10 00 10 00 P31 is Tx0 output, P30 is Rx0 input
35     ; Modes      31 30 21 20 P20 and 21 wired together for uart ck
36     ;
37     ;      P33 is interrupt on end of data
38     ; for      62 P2BH 01 10 00 10- P32 is Data ready to 8039, P22 is low if
39     ;      33 32 23 22 S8 has data, and P23 is rmi from emulator
50     ; Ports
40     ;
41     ;      08 P2CH 00 00 10 01
42     ;      35 34 25 24 HX0 on P24/25 conf for fast int?
43     ;
55

```

EP 0 388 560 A2

```

44      ;      80 P2DH 10 00 00 00 P37 enables 3695 Tx
45      ;      37 36 27 26 P27 is /ack from 8039 when in command
46      ;      mode
5
47
48      ; 241/0/PM defaults to good stuff.
49
50      0036 E6 F1 30      ld      PM,#X30
51
10      52      ; 240/0/PM default is OK, too.
53
54      0039 E6 F0 FF      ld      PM,#Xff
55
56      ; 252 & 253/0/P2xIP defaults OK
57      ; 246/0/P40 defaults to all inputs, which we want.
15      58      ; 247/0/P400 defaults to totem-pole, which we want.
59
60      ; 244/0/HOC : Handshake 0 used for Port 4, DMA to regs 0001 1101
61
62      003C E6 F4 00      ld      HOC,#X0
63
20      64      ; 245/0/H1C ...not used
65
66      003F E6 F0 FF      ld      POM,#XFF      ; programs rest of P0 as address lines
67      0042 E6 00 00      ld      X0,#X0
68      0045 E6 D8 FF      ld      SPH,#Xff      ; the high part of the stack pointer
25      69      0048 E6 D0 01      ld      IMR,#X1      ; interrupt mask reg, level 0 only
70      ; ok for 23 and 33
71      004B E6 FF 10      ld      IPR,#X10      ; interrupt priority register a>b>c
72      ; (port 23 mode set above as interrupt)
73
74      ; Counters
30      75
76      ; 224/0/COCT and 225/0/C1CT : idle
77      ; 224/1/COM and 225/1/C1H : simple timers
78
79      004E 5F      sb1
80      004F E6 E0 04      ld      COM,#4
35      81      0052 E6 E1 04      ld      C1H,#4.
82      0055 4F      sb0
83
84      ; initialize interrupt system
85
86      ; 255/0/IPR : default priorities OK.
40      87      ; 222/SYM : fast interrupts on IRQ4:
88
89      0056 E6 DE 10      ld      SYM,#X10      ;not needed yet, used dma
90
91      ; 221/IMR : enable IRQ4 for Handshake 0 later...
92
45      93      ; 254/0/EXT : defaults to stack in registers
94      ; ..so put it at top of RAM (in registers)
95
96      0059 80 FE      clr      EXT      ; fast memory; stack in registers
97      005B 80 D8      clr      SPH
50      98      005D E6 D9 FF      ld      SPL,#XFF
99
100     ; Program the UART:

```

EP 0 388 560 A2

```

101                                     ; /1/UMA : clock *32 rcv flag = 1 (16 desired by SP)
102                                     ; /1/UHB : use system clock xtal/2 to P2,
103
104 0060 5F                               S81
5 105 0061 E6 FE 00                       ld    UMUX,#20 ; don't user wake up mask
106 0064 E6 FF 00                       ld    UMSX,#20
107 0067 E6 FA 73                       ld    UMA,#273 ;0111 0011 ; div by 16 for 6 Mhz xtal,
108                                     ; no parity, wake up bits are
109                                     ; expected high for rec and sent
110                                     ; high on xmit.
11 111 006A E6 FB 44                       ld    UHB,#244 ;0100 0100
112 006D 4F                               S80
113
114                                     ; /0/UTC : set up and enable transmitter:
115
116 006E E6 EB BE                       ld    UTC,#2BE ; 10111110 12-bit chars, etc.
15 117                                     ; use P31 for xmit data, don't send
118                                     ; break, 2 stop bits, wake up enable!,
119                                     ; trans enable, zero count (not used),
120                                     ; TBE, and Trans DMA (off)
121
122                                     ; /0/UIE : no interrupts for now
20 123
124 0071 80 ED                       clr    UIE
125 0073 E6 EC 7C                       ld    URC,#27C ; 01111100 reset, but disabled
126
127                                     ; When ready to start, ld URC with #2 to enable receiver.
25 128 0076                               title IBM 4683 INTERFACE SOFTWARE FOR SUPER8
129
130 0000                               RSECT
131                                     ; Output buffer format:
132                                     ; dev_addr/SOLC/stat1/stat2/stat3/message
133                                     ; Device addr and status will remain here permanently.
30 134
135 0000                               out_buff ds 19 ; output buffer, starts at reg loc 0
136 0013                               out_len ds 1 ;
137 0014                               out_sdlc ds 1 ; 2nd byte of output msg
138 0015                               in_flag ds 1 ;
139 0016                               in_len ds 1 ; input length
35 140 0017                               in_buff ds 9 ; input buffer
141 0020                               our_stat ds 1 ; status byte:
142 0021                               RRcntr ds 1 ; RR_pend timeout counter
143
144 0000                               online equ 0 ; bit 0 = online
145 0001                               RR_pend equ 1 ; bit 1 = RR pending
40 146 0002                               SNRMed equ 2 ; bit 2 = SNRMed; owe him NSA
147
148 0022                               stat_req ds 1 ; 1 = send status on next cycle
149 0023                               EC_req ds 1 ; 1 = send EC on next cycle
150 0024                               set_RR ds 1
151
45 152 0002                               Sta_byte1 equ 2 ; offset from 0 for status bytes
153 0003                               Sta_byte2 equ 3 ; (adr is 0, 1 is SOLC byte)
154 0004                               Sta_byte3 equ 4 ; not used
155
156 0025                               delay1 ds 1
60 157 0026                               delay2 ds 1

```

55

IBM 4683 INTERFACE SOFTWARE FOR SUPER8

```

158 0027          ev11      ds      1          ;*****
159 0051          org      X51
5   160 0051          poll_ct ds      1          ;*   NOTE1
161              ;*   DON'T USE r1
162              ;*   (used by DMA)
163              ;*****
164 0076          CODE
165
10  166              ; Setup low register file:
167
168 0076 1C 50          ld      r1,#X50
169 0078 06 C1 00      slup    ld      r1,#X200
170 007B 1A FB          djnz   r1,slup
171 007D 80 00          clr     0
15  172 007F 0C 05      ld      r0,#X5
173 0081 80 C2          clr     r2
174 0083 C6 CC 04 26    ldw     rr12,#hdr
175 0087 E3 3C          hdlp:  ldci   r3,rr12      ; load up the above header in low reg space
176 0089 07 23          ld      r2,r3
20  177 008B 2E          inc     r2
178 008C 0A F9          djnz   r0,hdlp
179 008E 46 D3 04          or      r3,#X04      ; be sure p32 is high (0000 0100)
180
181              ;ROM checksum routine
182
25  183 0091 9C 00          chksum0: ld      r9,#0
184 0093 C6 CC 00 00      ldw     rr12,#0
185 0097 E3 8C          chksum1: ldci   r8,rr12      ; load next byte into reg
186              ; from program memory
187 0099 02 98          add     r9,r8      ; r9 to hold sum
188 009B A6 CD 00          cp      r13,#0      ; Address of last byte in EPROM
189              ; (low address byte)
30  190 009E 68 02          jr      eq,chksum2      ; done adding ?
191 00A0 88 F5          jr      chksum1      ; no, loop back
192 00A2 A6 CC 10          chksum2: cp      r12,#X10      ; Address of last byte in EPROM
193 00A5 E8 F0          jr      ne,chksum1      ; no, loop back
194 00A7 A6 C9 FF          cp      r9,#XFF      ; is checksum FF ?
35  195 00AA 68 02          jr      eq,chksum3      ; yes, jump
196 00AC 88 E3          jr      chksum0      ; repeat checksum until passed
197          chksum3:      ; good checksum
198
199 00AE 56 03 FE          and     Sta_byte2,#XFE      ; set scanner not alive,
40  200              ; status set to alive only
201              ; after successful handshake
202              ; with 8039
203
204              ; END OF INITIALIZTION
205 0081          uart_main:
206
45  207 0081 46 EC FE          or      URC,#XFE      ; Rx on
208 0084 E8 EF          ld      r14,U10      ; trash random input
209 0086 EC 17          ld      r14,#in_buff      ; point to buffer
210 0088 80 CC          clr     r12      ; count = 0
211 008A          poll2
50  212 008A 08 D2          ld      r0,p2      ; test 8039 PON line
213 008C 37 0F 00          btjrt  scnr_on,r0,#7      ; is the scanner on?
214 008F 38 03          ld      r3,Sta_byte2      ; no

```

IBM 4683 INTERFACE SOFTWARE FOR SUPER8

```

215 00C1 37 30 15          btjrf pon_ovr,r3,#0 ; is scanner already marked off?
216 00C4 56 03 FE          and Sta_byte2,#%fe ; no, mark scanner off bit
217 00C7 E6 22 01          ld stat_req,#%1 ; queue in an unrequested status
218 00CA 88 00              jr pon_ovr
219
220 00CC 38 03          scnr_on ld r3,Sta_byte2
221 00CE 37 31 08          btjrt pon_ovr,r3,#0 ; jmp, scanner alive bit already set
222 00D1 2C 77          ld r2,#%77 ; load valid checksum to send to 8039
223 00D3 F6 03 80          call tell_8039 ; send checksum result to 8039
224 ; successful handshake with 8039 will
225 ; mark scanner alive bit
226 00D6 E6 22 01          ld stat_req,#%1 ; queue in an unrequested status
227
228 ; SERVICE 4683
229
230 00D9          pon_ovr
231 00D9 88 EC          ld R8,URC
232 00DB 37 80 DC          btjrf poll2,r8,#0 ; loop until RDA
233 00DE
234 00DE 98 EF          poll3 ld R9,U10 ; get input
235 00E0 E6 EC FE          ld URC,#%FE ; reset UART recvr
236 00E3 37 8E D4          btjrf poll2,r8,#7 ; loop until MAD
237
238 00E6 37 9E 08          poll4 btjrf addresd,r9,#7 ; go if an address
239 00E9 A6 C9 CA          cp r9,#%CA ; 1010 1010 4a + 80
240 00EC E8 CC          jr ne,poll2 ; not my poll
241 00EE 8D 02 3D          jp polled ; ..else, polled
242 00F1
243 00F1 A6 C9 4A          addresd: cp r9,#%4A ; for me?
244 00F4 68 05          jr eq,me ; ..yes
245 00F6 A6 C9 7A          cp r9,#%7A ; broadcast?
246 00F9 E8 BF          jr ne,poll2 ; ..no - try again
247
248
249 00FB 76 C8 18          me: tm R8,#%18 ; A message for me!
250 00FE ED 01 74          jp nz,ERR01 ; any UART errors?
251 ; ..yes
252 0101 C6 CA FF FF          p_4_s ldw R210,#%FFFFFF ; init. CRC
253 0105 D7 E9          ld R14,R9 ; save input
254 0107 A6 CC 08          cp r12,#%08 ; limit in_buff to 9 bytes
255 010A 68 02          jr z,poll5
256 010C EE          inc r14
257 010D CE          inc r12 ; bump count
258
259 010E 08 EC          poll5 ld R0,URC
260 0110 37 00 F8          btjrf poll5,r0,#0 ; loop for RDA
261
262 0113 98 EF          poll6 ld R9,U10 ; get input
263 0115 A6 C9 7E          cp R9,#%7E ; flag (EOM)?
264 0118 E8 03          jr ne,poll7 ; ..no
265 011A 37 0F 06          btjrt crc0,R0,#7 ; go if 9th bit
266 011D 37 0E E5          poll7 btjrf p_4_s,R0,#7 ; error if 9th bit and not EOM
267 0120 8D 01 7E          jp ERR23 ; (needs a long jump)
268
269 ;***** EOM received *****
270
271 ; check CRC's

```

IBM 4683 INTERFACE SOFTWARE FOR SUPERS

```

272
5 273 0123 C9 16      crc0  ld    in_len,r12      ; stash count in buffer
274 0125 C6 CA FF FF      ldw   rr10,#FFFFF      ; init CRC
275 0129 EC 17          ld     r14,#in_buff
276 0128 C7 9E          crc1  ld     r9,r14
277 0120 F6 03 F7          call  checksum          ; do CRC check on byte
278 0130 EE            inc     r14              ; set to next byte
10 279 0131 CA F8          djnz   r12,crc1          ; length of message in r12
280
281 0133 A6 CA B8          cp     r10,#B8
282 0134 EB 41          jr      ne,ERR20
283 0138 A6 C8 F0          cp     r11,#F0
284 0138 EB 3C          jr      ne,ERR20          ; bad CRC
15 285 0130
286          ; check SDC counts...
287
288 013D 88 18          ld     r8,in_buff+1
289 013F 76 C8 11          tm     r8,#X11          ; is it RRRSSSD ?
290 0142 68 0C          jr      z,poll9          ; ..yes - msg
20 291 0144 98 C8          ld     r9,r8
292 0146 56 C9 1F          and    r9,#X1F
293 0149 A6 C9 01          cp     r9,#X01
294 014C 68 08          jr      eq,poll10          ; RR
295 014E 88 1F          jr      poll11          ; other SDC protocol bytes
25 296 0150
297 0150 56 C8 0E          and    r8,#X0E
298 0153 D0 C8          sra     r8
299 0155 A2 85          cp     r8,r5          ; vs my rrr
300 0157 EB 2A          jr      ne,ERR24
301 0159 88 18          poll10 ld    r8,in_buff+1
302 015B F0 C8          swap   r8
30 303 015D 56 C8 0E          and    r8,#X0E
304 0160 D0 C8          sra     r8
305 0162 A2 84          cp     r8,r4          ; vs my sss
306 0164 EB 22          jr      ne,ERR25
307 0166 88 18          ld     r8,in_buff+1
308 0168 37 81 04          btjrt  poll11,r8,#0
35 309 016A 5E            inc     r5              ; bump my RRR
310 016C 56 C5 07          and    r5,#7          ; RRR is 3 bits
311
312          poll11:          ; Have a good input message...
313
314 016F E6 15 FF          ld     in_flag,#XFF      ; flag = good msg
40 315 0172 88 19          jr      T0126          ; exit
316
317 0174 E6 15 01          ERR01 ld    in_flag,#X01      ; UART error
318 0177 88 14          jr      T0126
319
320 0179 E6 15 20          ERR20 ld    in_flag,#X20      ; CRC error
45 321 017C 88 0F          jr      T0126
322
323 017E E6 15 23          ERR23 ld    in_flag,#X23      ; improper 9th bit
324 0181 88 0A          jr      T0126
325
326 0183 E6 15 24          ERR24 ld    in_flag,#X24      ; his S < my R
50 327 0186 88 05          jr      T0126
328

```

55

IBM 4683 INTERFACE SOFTWARE FOR SUPER8

```

5      329 0188 E6 15 25      ERR25  ld    in_flag,#X25    ; his R <- my S
      330 0188 88 00          jr      T0126
      331
      332                      ; Common exit from this routine...
      333
      334 0180 56 EC FD      T0126  and    URC,#XFF-X02    ; stop UART receiver
      335
10     336
      337                      ; At this point, we have either:
      338                      ;   An error
      339                      ;   An input message
      340                      ;   A SNRM
      341                      ;   An RR
      342
15     343                      ; If we are offline, only a SNRM is acceptable; we should send
      344                      ; NSA, reset our counts, and mark ourselves on-line.
      345 0190 38 20          ld      r3,our_stat    ; get our status
      346                      ; (used throughout below code)
      347
20     348 0192 A6 15 FF      cp      in_flag,#XFF    ; did we get a message?
      349 0195 ED 00 B1      jp      ne,uart_main    ; ..nope
      350 0198 A6 18 B3      cp      in_buff+1,#X03    ; is it SNRM?
      351 019B EB 17          jr      ne,no_SNRM      ; ..no
      352
      353 019D 3C 32          ld      r3,#50          ; initialize RR timeout counter
25     354 019F 39 21          ld      RRcntr,r3
      355 01A1 3C 00          ld      r3,#X0          ; clear the world on a SNRM
      356 01A3 F6 04 28      call    mk_ready      ; could remove this part from main
      357 01A6 9F             ei                      ; where it leads into the uart call
      358
      359 01A7 77 35          bits    r3,#SNRMed      ; ..yes - mark it
30     360 01A9 77 31          bits    r3,#online     ; ..and put us online
      361 01AB 39 20          ld      our_stat,r3      ; save stat
      362
      363 01AD 80 C4          clr      r4              ; clear msg counts
      364 01AF 80 C5          clr      r5              ; ...
      365
35     366 01B1 8D 00 B1      jp      uart_main      ; and exit
      367
      368                      ; If we just sent a message, then we only expect an RR.
      369
      370 01B4 37 32 15      no_SNRM btjrf no_RR,r3,RRR_pend ; go if no RR reqd
40     371 01B7 28 18          ld      r2,in_buff+1    ; get SDLC byte
      372 01B9 56 C2 1F      and      r2,#X1F        ; should be RRR00001
      373 01BC A6 C2 01      cp      r2,#1
      374 01BF EB 08          jr      ne,no_RR        ; ..but it isnt
      375 01C1 77 32          bitr     r3,RRR_pend      ; reset RR pending bit
      376 01C3 39 20          ld      our_stat,r3      ; save status
45     377 01C5 F6 04 28      call    mk_ready      ; scans from the 8039
      378 01C8 9F             ei
      379 01C9 8D 00 B1      jp      uart_main      ; and exit
      380
      381 01CC 37 31 03      no_RR  btjrt  we_is_on,r3,#online ; go if online
50     382 01CF 8D 00 B1      jp      uart_main      ; ERROR - no valid msgs here
      383
      384 01D2
      385 01D2 28 18          we_is_on: ld      r2,in_buff+1    ; get SDLC byte

```

55

EP 0 388 560 A2

IBN 4683 INTERFACE SOFTWARE FOR SUPER8

```

386 0104 56 C2 11      and    r2,#X11      ; should be RRR0SS$0
387 0107 ED 00 B1      jp      nz,uart_main ; ..but it isn't
5
388
389      ; Got a good message...process it
390
391 01DA E6 24 01      ld      set_RR,#X1      ; flag that next poll gets RR
392
10
393 01DD 28 19      ld      r2,in_buff+2      ; get command byte
394 01DF A6 C2 00      cp      r2,#X200
395 01E2 68 39      jr      z,sys_cmd      ; go if system command
396
397 01E4 A6 C2 11      cp      r2,#X11      ; enable command
398 01E7 E8 09      jr      ne,cmd1
15
399 01E9 46 03 02      or      sta_byte2,#X2      ; 0000 0010
400 01EC F6 03 80      call    tell_8039
401 01EF 8D 00 B1      jp      uart_main
402
403 01F2 A6 C2 12      cmd1 cp      r2,#X12
404 01F5 E8 09      jr      ne,cmd2      ; disable
20
405 01F7 56 03 FD      and     sta_byte2,#Xfd      ; 1111 1101
406 01FA F6 03 80      call    tell_8039
407 01FD 8D 00 B1      jp      uart_main
408
409 0200 A6 C2 14      cmd2 cp      r2,#X14
410 0203 E8 09      jr      ne,cmd3      ; beep enable
25
411 0205 46 02 10      or      sta_byte1,#X10      ; 0001 0000
412 0208 F6 03 80      call    tell_8039
413 0208 8D 00 B1      jp      uart_main
414
415 020E A6 C2 18      cmd3 cp      r2,#X18
416 0211 ED 00 B1      jp      ne,uart_main      ; disable-beep
30
417 0214 56 02 EF      and     sta_byte1,#Xef      ; 1110 1111
418 0217 F6 03 80      call    tell_8039
419 021A 8D 00 B1      jp      uart_main
420
421 021D 28 1A      sys_cmd ld    r2,in_buff+3      ; get system command
422
35
423 021F 37 28 0F      btjrt   stat_set,r2,#5      ; status req
424 0222 37 2F 12      btjrt   EC_set,r2,#7      ; EC# req
425 0225 37 29 09      btjrt   stat_set,r2,#4      ; test req = status req
426 0228 37 2D 03      btjrt   sys_reset,r2,#6      ; system reset
427
428 022B 8D 00 B1      jp      uart_main      ; bad system command
40
429
430 022E      sys_reset:
431 022E 8D 00 20      jp      X0020      ; restart program!
432
433 0231      stat_set:
434 0231 E6 22 01      ld      stat_req,#X1      ; send status on next poll
45
435 0234 8D 00 B1      jp      uart_main
436
437 0237      EC_set:
438 0237 E6 23 01      ld      EC_req,#X1      ; send EC on next poll cycle
439 023A 8D 00 B1      jp      uart_main
50
440
441      ; We got 1st poll character...look for 2nd
442

```

55

IBM 4683 INTERFACE SOFTWARE FOR SUPER8

```

5      443 0230 88 EC      polled ld    R8,URC
      444 023F 37 B0 FB      btjrf  polled,R8,#0      ; loop until RDA
      445 0242 98 EF      ld      r9,U10      ; get input
      446 0244 E6 EC FF      ld      URC,$2FF      ; clear status
      447 0247 A6 C9 CA      cp      r9,$2CA
      448 024A 68 03      jr      eq,poll_4me
      449
10     450 024C 80 00 BA      jp      poll2      ; else, ignore
      451
      452 024F      poll_4me:
      453
      454      ; If offline, we will always send ROL.
      455      ; If SNRMd, we will send NSA.
      456      ; If out_len < 0, send message from out_buff.
      457      ; If out_len = 0, simply send EOP.
      458
      459 024F      ok:
      460 024F 38 20      ld      r3,our_stat
      461 0251 37 31 06      btjrt  p_4a,r3,#online ; go if on_line
      462 0254 F6 03 C7      call   SEND_ROL      ; send ROL
      463 0257 80 00 B1      jp      uart_main      ; ..and exit
      464
      465 025A 37 34 00      p_4a  btjrf  p_4b,r3,#SNRMd ; go if not SNRMd
      466 0250 F6 03 CC      call   SEND_NSA      ; if SNRM, send NSA
      467 0260 77 34      bitr   r3,#SNRMd      ; reset request for NSA
      468 0262 39 20      ld      our_stat,r3      ; ..in rsect
      469 0264 E6 22 01      ld      stat_req,$21      ; queue an unrequested status
      470 0267 80 00 B1      jp      uart_main      ; ..and exit
      471 026A
      472 026A A6 24 00      p_4b:  cp      set_RR,$20      ; check if RR is needed
      473 0260 60 02 79      jp      z,p_4bb
      474 0270 E6 24 00      ld      set_RR,$20      ; clear flag
      475 0273 F6 03 B1      call   SEND_RR
      476 0276 80 00 B1      jp      uart_main
      477
      478 0279 A6 22 00      p_4bb: cp      stat_req,$20      ; see if we need to send status
      479 027C 68 16      jr      z,p_4c      ; skip if no request for status
      480 027E E6 01 00      ld      out_buff+1,$20      ; need to wipe the SDLC byte
      481 0281 EC 00      ld      r14,$20      ; msg pointer
      482 0283 CC 05      ld      r12,$25      ; length = 5
      483 0285 E6 22 00      ld      stat_req,$20      ; clear status request
      484 0288 F6 02 F8      call   MSG_OUT
      485 0288 38 20      ld      r3,our_stat
      486 0280 77 33      bits   r3,$RR_pend      ; mark RR pending
      487 028F 39 20      ld      our_stat,r3
      488 0291 80 00 B1      jp      uart_main
      489
      490 0294
      491 0294 A6 23 00      p_4c:  cp      EC_req,$20      ; see if an EC req has been queued
      492 0297 68 1F      jr      z,p_4d      ; again skip if no EC request
      493 0299 E6 01 00      ld      out_buff+1,$20      ; must wipe SDLC byte or MSG_OUT fails
      494 029C 46 02 01      or      out_buff+2,$201      ; flag that EC present
      495 029F E6 05 04      ld      out_buff+5,$204      ; the observed EC #
      496 02A2 EC 00      ld      r14,$out_buff      ; EC might overlay first char in msg
      497 02A4 CC 06      ld      r12,$26      ; length = 6
      498 02A6 E6 23 00      ld      EC_req,$20      ; clear EC request
      499 02A9 F6 02 F8      call   MSG_OUT

```

IBM 4683 INTERFACE SOFTWARE FOR SUPER8

5

10

15

20

25

30

35

40

45

50

55

```

500 02AC 56 02 FE      and    out_buff+2,%fe ; clear EC present from status
501 02AF 38 20      ld      r3,our_stat
502 02B1 77 33      bits    r3,RRR_pend ; mark RR pending
503 02B3 39 20      ld      our_stat,r3
504 02B5 80 00 B1    jp      uart_main
505
506 02B8 A6 13 00    p_4d:  cp      out_len,#0 ; last check for msgs to go
507 02B8 E8 06      jr      nc,p_4e ; there is a msg to go
508 02B0 F6 03 D1    call    SEND_EOP ; else, send EOP
509 02C0 80 00 B1    jp      uart_main
510
511 ; may have already sent msg and could be awaiting RR
512
513 02C3      p_4e:
514 02C3 38 20      ld      r3,our_stat
515 02C5 37 32 16    btjrf  p_4f,r3,RRR_pend ; test to keep msg from dup xmits
516 02C8 70 C9      push    r9
517 02CA 98 21      ld      r9,RRcntnr ; RR timeout counter
518 02CC 00 C9      dec     r9
519 02CE A6 C9 00    cp      r9,#0
520 02D1 E8 02      jr      nz,p_4ee ; if RR timeout, mark ourselves
521 02D3 77 30      bitr   r3,#online ; offline
522 02D5 99 21      p_4ee: ld      RRcntnr,r9
523 02D7 50 C9      pop     r9
524 02D9 39 20      ld      our_stat,r3
525 02DB 80 00 B1    jp      uart_main
526
527 02DE CC 32      p_4f:  ld      r12,#50 ; reset RR timeout counter
528 02E0 C9 21      ld      RRcntnr,r12 ; get ready to send label
529 02E2 C8 13      ld      r12,out_len ; get length
530 02E4 EC 00      ld      r14,#out_buff ; ..and address
531 02E6 EE          inc     r14 ; point to SDLC byte
532 02E7 06 CE 00    ld      @r14,#20 ; clear byte (see MSG_OUT for why)
533 02EA 00 CE      dec     r14 ; point back to start
534 02EC F6 02 F8    call    MSG_OUT ; send msg
535 02EF 38 20      ld      r3,our_stat
536 02F1 77 33      bits    r3,RRR_pend ; mark RR pending
537 02F3 39 20      ld      our_stat,r3
538 02F5 80 00 B1    jp      uart_main ; exit
539
540 ; Message output... assumes register usage:
541 ; R14 = address of msg to send (RSECT)
542 ; R12 = length of message (max. 20 )
543 ; RR10 : will be used for checksums
544 ; R9 : will carry one byte for output
545 ; R8 : will carry flags for output
546 ; RR4 : SDLC counts
547 02FB      MSG_OUT:
548 02FB 56 EC FD      and    URC,%XFF-X02 ; receiver off
549 02FB 46 D3 80      or     P3,%X80 ; P37 on (3695 xmitter)
550 02FE 0C 0C      ld      R0,#12 ;
551 0300 0A FE      lll1:  djnz   R0,lll1 ; delay for awhile 10
552 0302 5F          sb1    ; 6
553 0303 46 FA 01    or     UMA,%X01 ; 9th bit high (1 state) 10
554 0306 4F          sb0    ; 6
555 0307 C7 9E      ld      R9,@R14 ; get 1st byte 10
556 0309 EE          inc     R14 ; 6

```

EP 0 388 560 A2

IBM 4683 INTERFACE SOFTWARE FOR SUPER8

```

5
557 030A 99 EF          ld      U10,R9          ; ..send it          10
558                                ;               need 380-388 for address
559 030C C6 CA FF FF      ldw     R8,R9,0xFFFF    ; Init. CRC          12
560 0310 F6 03 F7          call    chksum          ; 18+166
561
10
562                                ; set SDLC counts:
563 0313 C7 8E          ld      R8,R14          ; get byte          6
564 0315 76 C8 11          tm      R8,R11          ; normal msg?      10
565 0318 E8 0C          jr      nz,MOX          ; ..no - protocol  12
566
15
567 031A 88 C5          ld      R8,R5          ; SDLC rrr          6
568 031C F0 C8          swap     R8              ;               6
569 031E 42 84          or      R8,R4          ; SDLC sss          6
570 0320 90 C8          rl      R8              ;               6
571 0322 D7 E8          ld      R14,R8          ; stuff counts      6
572 0324 88 07          jr      M_O_0          ;               12
573
20
574 0326 FF          MOX     nop
575 0327 FF          nop
576 0328 FF          nop
577 0329 FF          nop
578 032A FF          nop
579 032B 88 00          jr      M_O_0          ;               12
580
25
581 032D 89 14          M_O_0: ld      out_adlc,R8    ; remember it      10
582 032F 5F          sb1          ;               6
583 0330 56 FA FE          and     U1A,0XFF-201    ; 9th bit 0 state  10
584 0333 4F          sb0          ;               6
30
585 0334 00 CC          M_O_1: dec     R12          ;               6
586 0336 68 0A          jr      z,M_O_2          ; go if no more data 10
587 0338 C7 9E          ld      R9,R14          ; get byte          6
588 033A EE          inc      R14          ;               6
589
35
590 033B 99 EF          ld      U10,R9          ; (192 2nd) total 384 10
591
592 033D F6 03 F7          call    chksum          ;               12+166
593 0340 88 F2          jr      M_O_1          ;               12
594
40
595 0342 98 CA          M_O_2: ld      R9,R10          ; get 1st chksum    6
596 0344 89 27          ld      sv11,R11        ; ..save 2nd        6
597 0346 60 C9          com      R9              ; compl. 1st one    6
598 0348 99 EF          ld      U10,R9          ; ..send it (close) 10
599
45
600 034A F6 03 F7          call    chksum          ; ..and add to CRC  178
601 034D 98 27          ld      R9,sv11        ; get original 2nd CRC 6
602 034F 60 C9          com      R9              ;               6
603 0351 99 EF          ld      U10,R9          ; send it total?   10
604
50
605 0353 F6 03 F7          call    chksum          ;               178
606 0356 9C 7E          ld      R9,0X7E        ; load flag byte    10
607 0358 5F          sb1          ;               6
608 0359 46 FA 01          or      U1A,0X01        ; 9th bit 1 state  10
609 035C 4F          sb0          ;               6
610 035D 0C 0F          ld      r0,R15          ;               10
611 035F 0A FE          MOXCS djnz    r0,MOXCS    ;               152
612 0361 99 EF          ld      U10,R9          ; xmit              10
613
55

```

EP 0 388 560 A2

IBM 4683 INTERFACE SOFTWARE FOR SUPERS

```

614 0363 9C 15          ld      r9,#21
615 0365 9A FE          M_0_3a djnz  r9,M_0_3a
5
616
617 0367 56 D3 7F      and     P3,#27F      ; 3695 Tx off
618 036A 88 EF          ld      r8,U10
619 036C 46 EC FE      or      URC,#2FE      ; UART Rx on
620 036F 5F            sb1
10
621 0370 56 FA FE      and     UMA,#2FF-201      ; 9th bit 0 state
622 0373 4F            sb0
623 0374 88 14          ld      R8,out_sdlc      ; get SDLC byte
624 0376 76 E8 11      tm      R8,#211      ; was it protocol?
625 0379 EB 04          jr      nz,M_0_4      ; ..yes
626 037B 4E            inc     r4      ; inc sss
15
627 037C 56 C4 07      and     r4,#7      ; sss is 3 bits
628 037F AF          M_0_4 ret      ; exit routine
629
630 tell_8039:
631 0380 E6 22 01      ld      stat_req,#21      ; RR will go out on next poll,
632          8000      ls374 equ     28000      ; send status on poll after that
20
633 0383 87 21 00 80      lde     ls374,r2      ; anything beyond rom will do
634 0387 C6 C6 00 64      ldw     rr6,#100      ; stuff command byte into buffer chip
635 0388 56 D3 FB      and     P3,#2FB      ; adjustable timeout for 8039 to respond
636 038E 80 C6      stloc: decw  rr6      ; bit 2 on port 3 (P27 on 8039)
637 0390 68 1E          jr      z,give_up
638 0392 08 D2          ld      r0,P2
25
639 0394 56 C0 08      and     r0,#208      ; sample
640 0397 A6 C0 00      cp      r0,#20      ; 0000 1000
641 039A EB F2          jr      nz,$loc      ; P12 will go low and stay that way
642          ;
643 039C 46 D3 04      or      P3,#204      ; indicating data has been read.
644          ;
30
645 039F 80 C6      stloc2 decw  rr6      ; When found low,
646 03A1 68 0D          jr      z,give_up      ; 0000 0100 raise line to 8039
647 03A3 08 D2          ld      r0,P2
648 03A5 56 C0 08      and     r0,#208
649 03A8 A6 C0 00      cp      r0,#20
35
650 03AB 68 F2          jr      z,$loc2
651 03AD 46 D3 01      or      Sta_byte2,#201 ; successful communication to 8039 so
652          ; mark scanner alive bit
653
654
655 0380 AF          give_up: ret      ; time out
656
40
657          ; ***** send Receive-Ready *****
658 0381
SEND_RR:
659 0381 88 C5          ld      r8,r5      ; my RRR
660 0383 F0 C8          swap   r8
661 0385 90 C8          rl      r8
45
662 0387 46 C8 01      or      r8,#1
663 038A 89 01          ld      out_buff+1,r8
664 038C
common_short:
665 038C E6 00 4A      ld      out_buff,#24A      ; my addr
666 038F EC 00          ld      r14,#out_buff      ; point to buffer
667 03C1 CC 02          ld      r12,#2      ; count = 2
60
668 03C3 F6 02 FB      call   MSC_OUT
669 03C6 AF          ret
670

```

EP 0 388 560 A2

IBM 4683 INTERFACE SOFTWARE FOR SUPER8

	671			; ***** Send ROL *****
5	672	03C7		SEND_ROL:
	673	03C7	E6 01 0F	ld out_buff+1,#X0F ; ROL
	674	03CA	88 FD	jr common_short
	675			
	676			;***** Send NSA *****
10	677	03CC		SEND_NSA:
	678	03CC	E6 01 63	ld out_buff+1,#X63 ; NSA
	679	03CF	88 EB	jr common_short
	680			
	681			;***** Send EOP *****
	682	03D1		SEND_EOP:
15	683	03D1	56 EC FD	and URC,#XFF-X02 ; receiver off
	684	03D4	46 D3 80	or P3,#X20 ; P37 on (3695 xmitter) 195 to xmit
	685	03D7	0C DE	ld R0,#14 ; NOT hex # 10 cyc
	686	03D9	0A FE	lll7 djnz R0,lll7 ; delay for awhile n*10 (12 last)
	687	03DB	FF	nop ; 6
	688	03DC	5F	sbl ; 6
20	689	03DD	46 FA 01	or LMA,#X01 ; 9th bit 1 state 10
	690	03ED	4F	sbl ; 6
	691			
	692	03E1	E6 EF 5A	ld U10,#X5A ; send EOP 10 total 192
	693			
	694	03E4	DC 11	ld r0,#17 ; need 195 cycles 10
25	695	03E6		Sloc_del:
	696	03E6	0A FE	djnz r0,Sloc_del ; n*12
	697	03E8	FF	nop ;
	698	03E9	56 D3 7f	and P3,#X7F ; 3695 Tx off total 198 10
	699	03EC	88 EF	ld r8,U10 ; trash input?
	700	03EE	46 EC FE	or URC,#XFE ; UART Rx on
30	701	03F1	5F	sbl
	702	03F2	56 FA FE	and LMA,#XFF-X01 ; 9th bit 0 state
	703	03F5	4F	sbl ; 12/24/B6
	704	03F6	AF	ret
	705			
	706			
35	707			;
	708			; **** Checksum routine *****
	709			; Uses RR10 for chksums.
				; Current byte in R9.
	710	03F7	70 C3	chksum: push r3 ; 10
	711	03F9	B2 9A	xor R9,R10 ; 6
	712	03FB	D8 C9	ld R0,R9 ; 6
40	713	03FD	F0 C9	swap R9 ; 8
	714	03FF	56 C9 F0	and R9,#XF0 ; 10
	715	0402	B2 90	xor R9,R0 ; 6
	716	0404	38 C9	ld R3,R9 ; 6
	717	0406	F0 C9	swap R9 ; 6
	718	0408	56 C9 DF	and R9,#XDF ; 10
45	719	040B	B2 98	xor R9,R11 ; 6
	720	040D	A8 C9	ld R10,R9 ; 6
	721	040F	98 C3	ld R9,R3 ; 6
	722	0411	B8 C9	ld R11,R9 ; 6
	723	0413	F0 C9	swap R9 ; 6
	724	0415	E0 C9	rr R9 ; 6
50	725	0417	08 C9	ld R0,R9 ; 6
	726	0419	56 C0 07	and R0,#X07 ; 10
	727	041C	56 C9 FB	and R9,#XFB ; 10

EP 0 388 560 A2

IBM 4683 INTERFACE SOFTWARE FOR SUPER8

```

5      728 041F B2 A9          xor    R10,R9          ; 6
      729 0421 B2 B0          xor    R11,R0          ; 3
      730 0423 50 C3          pop     r3              ; 10
      731 0425 AF             ret                    ; 14
      732
      733 0426 4A 00 10 01 00  hdr:  db    %4a,x00,x10,x01,x00
      734
      735                                ; device address, SDLC byte,
      736                                ; and default status bytes (enabled)
10     737 0428              mk_ready:
      738 0428 46 D2 20          or     P2,%x20        ; be sure P25 is high state
      739                                ; 0010 0000
      740 042E 46 D3 04          or     P3,%x04        ; also p32 high (0000 0100)
      741                                ;
15     742 0431 5F             sb1
      743 0432 E6 F0 00          ld     DCH,%x0        ; DMA high byte
      744 0435 E6 F1 20          ld     DCL,%x20        ; DMA low byte, will int when zero
      745 0438 4F             sb0
      746
20     747 0439 1C 05          ld     r1,%x25          ; used to point into register file, by DMA
      748                                ; on int will hold msg length ->out_len
      749 0438 E6 13 00          ld     out_len,%x0      ; clear semaphore to communication side
      750 043E E6 F4 10          ld     NOC,%x1d        ; load/reload for handshake operation.
      751                                ; note-do not split this command!
      752 0441 AF             ret
25     753
      754              int_p33:
      755 0442 4F             sb0
      756 0443 E6 D6 C0          ld     RPO,%x2C0        ; hdu derived from msb on last byte
      757 0446 E6 F4 00          ld     NOC,%x0        ; set register bank pointer
      758 0449 56 D2 DF          and    P2,%x2f        ; note, r1 will contain # of char+1
      759 044C E6 FC FF          ld     P2AIP,%x2ff      ; turn off handshake and set T0 low
30     760                                ; 1101 1111 P25 = T0
      761                                ; reset int pending reg (see 8.7.4)
      762                                ; in super8 book
      763                                ; Have EOT. r1 is size of output msg. (buff must be 0 0)
      764                                ; semaphore to the
      765                                ; UART code via out_len. (EOT not part of msg)
35     766 044F 00 C1          dec     r1              ; compensates for EOT byte
      767 0451 19 13          ld     out_len,r1          ; semaphore to UART code (out_len ne 0)
      768                                ; uart code will find this;set and
      769                                ; xmit
40     770 0453 8F             ired
      771
      772              ;CODE CHECKSUM
      773 0454 C0              DB      CDH
      774              ;DATE
      775 0455 03 23 B8          DB      03H,23H,B8H
      776              ;PART NUMBER
45     777 0458 52 96 01 38          DB      'R',96H,01,59
      778              ;REV
      779 045C 41              DB      'A'
      780
      781 0450              vec_0
60     782 0450 6C 00          ld     r6,%x0
      783 045F              vec_2
      784 045F 6C 02          ld     r6,%x2
      785
      786
      787
      788
      789
      790
      791
      792
      793
      794
      795
      796
      797
      798
      799
      800
      801
      802
      803
      804
      805
      806
      807
      808
      809
      810
      811
      812
      813
      814
      815
      816
      817
      818
      819
      820
      821
      822
      823
      824
      825
      826
      827
      828
      829
      830
      831
      832
      833
      834
      835
      836
      837
      838
      839
      840
      841
      842
      843
      844
      845
      846
      847
      848
      849
      850
      851
      852
      853
      854
      855
      856
      857
      858
      859
      860
      861
      862
      863
      864
      865
      866
      867
      868
      869
      870
      871
      872
      873
      874
      875
      876
      877
      878
      879
      880
      881
      882
      883
      884
      885
      886
      887
      888
      889
      890
      891
      892
      893
      894
      895
      896
      897
      898
      899
      900
      901
      902
      903
      904
      905
      906
      907
      908
      909
      910
      911
      912
      913
      914
      915
      916
      917
      918
      919
      920
      921
      922
      923
      924
      925
      926
      927
      928
      929
      930
      931
      932
      933
      934
      935
      936
      937
      938
      939
      940
      941
      942
      943
      944
      945
      946
      947
      948
      949
      950
      951
      952
      953
      954
      955
      956
      957
      958
      959
      960
      961
      962
      963
      964
      965
      966
      967
      968
      969
      970
      971
      972
      973
      974
      975
      976
      977
      978
      979
      980
      981
      982
      983
      984
      985
      986
      987
      988
      989
      990
      991
      992
      993
      994
      995
      996
      997
      998
      999

```

IBM 4683 INTERFACE SOFTWARE FOR SUPER8

```

5      785 0461          vec_4
      786 0461 6C 04      ld      r6,#X4
      787 0463          vec_6
      788 0463 6C 06      ld      r6,#X6
      789 0465          vec_8
      790 0465 6C 08      ld      r6,#X8
      791 0467          vec_c
      792 0467 6C 0C      ld      r6,#XC
10     793 0469          vec_e
      794 0469 6C 0E      ld      r6,#XE
      795 0468          vec_10
      796 0468 6C 10      ld      r6,#X10
      797 0460          vec_12
15     798 0460 6C 12      ld      r6,#X12
      799 046F          vec_14
      800 046F 6C 14      ld      r6,#X14
      801 0471          vec_16
      802 0471 6C 16      ld      r6,#X16
      803 0473          vec_18
20     804 0473 6C 18      ld      r6,#X18
      805 0475          vec_1a
      806 0475 6C 1A      ld      r6,#X1a
      807 0477          vec_1c
      808 0477 6C 1C      ld      r6,#X1c
25     809 0479          vec_1e
      810 0479 6C 1E      ld      r6,#X1e
      811 047B 7C FF      ld      r7,#Xff
      812 047D 79 FC      ld      P2AIP,r7
      813 047F 79 FD      ld      P2BIP,r7
      814 0481 8F          ired
30     815
      816
      817 0482          ends
      817 0482          end
      817 0482          end

```

; to clear int pending reg.

Defined	Symbol Name	Value	References													
5	Pre CODE	0000	162	164	816											
	Pre DATA	0000														
	149 EC_req	0023	438	491	498											
	437 EC_get	0237	424													
	317 ERR01	0174	250													
10	320 ERR20	0179	282	284												
	323 ERR23	017E	267													
	326 ERR24	0183	300													
	329 ERR25	0188	306													
	574 MOX	0326	565													
15	611 MOXS	035F	611													
	547 MSG_OUT	02F8	484	499	534	668										
	581 H_O_0	0320	572	579												
	585 H_O_1	0334	593													
	595 H_O_2	0342	586													
20	615 H_O_3a	0365	615													
	628 H_O_4	037F	625													
	145 RR_pend	0001	370	375	486	502	515	536								
	142 RRcntr	0021	354	517	522	528										
	Pre RSECT	0000	130	159												
25	682 SEND_EOP	0301	508													
	677 SEND_RSA	03CC	466													
	672 SEND_RDL	03C7	462													
	658 SEND_RR	03B1	475													
	146 SHRMcd	0002	359	465	467											
30	152 Sta_byte1	0002	411	417												
	153 Sta_byte2	0003	199	214	216	220	399	405	651							
	154 Sta_byte3	0004														
	334 T0126	0180	315	318	321	324	327	330								
	242 addred	00F1	238													
35	183 chksum0	0091	196													
	185 chksum1	0097	191	193												
	192 chksum2	00A2	190													
	197 chksum3	00AE	195													
	710 chksum	03F7	277	560	592	600	605									
40	403 cmd1	01F2	398													
	409 cmd2	0200	404													
	415 cmd3	020E	410													
	664 common_short	038C	674	679												
	273 crc0	0123	265													
45	276 crc1	0128	279													
	156 delay1	0025														
	157 delay2	0026														
	10 dummy	0000														
	654 give_up	0380	637	646												
50	175 hdlp	0087	178													
	733 hdr	0426	174													
	140 in_buf1	0017	209	275	288	301	307	350	371	385	393	421				
	138 in_flag	0015	314	317	320	323	326	329	348							
	139 in_len	0016	273													
55	754 int_p33	0442	13													
	551 l111	0300	551													
	686 l117	0309	686													
	632 la374	8000	633													
	248 mc	00F8	244													
45	737 mk_ready	0428	356	377												

EP 0 388 560 A2

	Defined	Symbol Name	Value	References															
5	381	no_RR	01CC	370	374														
	370	no_SRRM	0184	351															
	459	ok	024F																
	144	online	0000	360	381	461	521												
	141	our_stat	0020	345	361	376	460	468	485	487	501	503	514	524					
10	135	out_buff	0000	535	537														
				480	493	494	495	496	500	530	663	665	666	673					
				678															
	136	out_len	0013	506	529	749	767												
	137	out_sdlc	0014	581	623														
15	253	p_4_a	0105	266															
	465	p_4a	025A	461															
	471	p_4b	026A	465															
	478	p_4bb	0279	473															
	490	p_4c	0294	479															
20	506	p_4d	0288	492															
	513	p_4e	02C3	507															
	522	p_4ee	02D5	520															
	527	p_4f	02DE	515															
	301	poll10	0159	294															
25	312	poll11	016F	295	308														
	211	poll12	008A	232	236	240	246	450											
	234	poll13	000E																
	238	poll14	00E6																
	259	poll15	010E	255	260														
30	262	poll16	0113																
	266	poll17	011D	264															
	296	poll19	0150	290															
	432	poll_4me	024F	448															
	160	poll_ct	0051																
35	443	polled	023D	241	444														
	230	pon_ovr	0009	215	218	221													
	220	scnr_on	00CC	213															
	150	set_RR	0024	391	472	474													
	17	start	0020																
40	148	stat_req	0022	217	226	434	469	478	483	631									
	433	stat_set	0231	423	425														
	158	sv11	0027	596	601														
	421	sys_cmd	021D	395															
	430	sys_reset	022E	426															
45	630	tell_8039	0380	223	400	406	412	418											
	205	uart_main	0081	349	366	379	382	387	401	407	413	416	419	428					
				435	439	463	470	476	488	504	509	525	538						
	781	vec_0	0450	12															
	795	vec_10	0468	14															
50	797	vec_12	0460	14															
	799	vec_14	046F	14															
	801	vec_16	0471	14															
	803	vec_18	0473	15															
	805	vec_1a	0475	15															
55	807	vec_1c	0477	15															
	809	vec_1e	0479	15															
	783	vec_2	045F	12															
	785	vec_4	0461	12															
	787	vec_6	0463	12															
60	789	vec_8	0465	13															
	791	vec_c	0467	13															

EP 0 388 560 A2

Defined	Symbol Name	Value	References
793	vec_e	0469	13
384	uc_is_on	01D2	381

5

Lines Assembled : 817

Assembly Errors : 0

10

The control software stored in interface memory means 218 may be exemplified by the following listing.

15

20

25

30

35

40

45

50

55

ISIS-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
GMA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 1

```

5      LOC OBJ      LINE      SOURCE STATEMENT
      1 ;
      2 ; FILE: GMA095.SRC 04-16-87      DREW TAUSSIG
      3 ;
      4 ; FIRMWARE FOR THE 750SL SCANNER
      5 ; WILL READ UPC-A,E AND EAN/JAN8,13
10     6 ; IBM-4683 SERIAL I/O CHANNEL, I/F OPTION #95.
      7 ;
      8 ; S-P PART NUMBER R96-0153
      9 ;
10     10 ;
11     11 $ INCLUDE(:F1:NISTRY.SRC)
12     12 ;
13     13 ; FILE: NISTRY.SRC 01-23-87 11:50 BOB ACTIS
14     14 ;
15     15 ; IVRD53 - 750F IG DEMO, SP-DC, VERSION 03 W/ 2ND CHECK      MAY 1983
16     16 ;
17     17 ; FVRD53 - 750F FT DEMO, SP-DC, VERSION 03 W/ 2ND CHECK      OCT 1983
18     18 ;
19     19 ; GMA053 - 750SL, SP-DC/MCR-OCIA, READS ALL UPC,EAN & JAN      JUN 1986
20     20 ;
21     21 ; GMA069 - 750SL, SMEDA PARALLEL/DMRON, READS ONLY A,E,8,13    OCT 1986
22     22 ;
23     23 ; GMA066 - 750SL, IBM-OCR/FUJITSU                                OCT 1986
24     24 ; IBM-OCR READS ONLY A,E,8,13
25     25 ; FUJITSU WILL ALSO READ VERSION D
26     26 ;
27     27 ; IFTP95 - 750F, IBM4683 SERIAL I/O CHANNEL, ONLY A,E,8,13    APR 1987
28     28 ;
29     29 ;
30     30 $ INCLUDE(:F1:FTDEFS.SRC)
31     31 ;
32     32 ; FILE: FTDEFS.SRC 12-09-86 13:50 BOB ACTIS
33     33 ;
34     34 ; *** SYSTEM DEFINITION ***
35     35 ;
36     36 ; PORT 1 DEFINITION:
37     37 ;
38     38 ETEST EQU 00000001B ;P10-0-ENABLE TEST MODE
39     39 ;
40     40 ETHARK EQU 00000010B ;P10-0-I/F RESET
41     41 ENTREB EQU 00000010B ;P11-0-TEST MARK
42     42 EP12 EQU 00000100B ;P11-0-ENABLE MOTOR
43     43 EP13 EQU 00001000B ;P12-0-HANDSHAKE FOR SUPER-8 TO 8039 I/F
44     44 ELASDB EQU 00010000B ;P13-1-MODE CONTROL FOR SB-8039 I/F
45     45 ECDLT EQU 00100000B ;P14-0-DISABLE LASER
46     46 EBDLT EQU 01000000B ;P15-0-ENABLE GOOD LIGHT
47     47 ETONE EQU 10000000B ;P16-0-ENABLE BAD LIGHT
48     48 ;
49     49 ; PORT 2 DEFINITION:
50     50 ;
51     51 EQU 00001111B ;P20-P23 EXTERNAL PROGRAM ADDRESS LINES
52     52 ESENT EQU 00010000B ;P24-1-VLSI DATA SENT* (USED ONLY FOR TEST)
53     53 EUP2SP EQU 00100000B ;P25-1-MOTOR UP2SPD SIGNAL (750SL ONLY)
54     54 EVLSIR EQU 01000000B ;P26-0-VLSI POWER RESET* (750SL ONLY)
55     55 EDISHS EQU 10000000B ;P27-1-HANDSHAKE FOR SUPER-8 TO 8039 I/F
56     56 ;
57     57 ; ;T0-1-HANDSHAKE FOR 8039 TO SUPER-8 I/F
58     58 ;
59     59 ;
60     60 ; FRAME CONTROL ARRAY:
61     61 ;
62     62 ; EXTERNAL MEMORY ADDRESSES
63     63 ;
64     64 EPARRD EQU 00H ;R - PARITY BYTE
65     65 ESRRD EQU 01H ;R - SEGMENT REGISTER
66     66 EFRST EQU 01H ;W - FRAME RESET (CLEARS SEGMENT)
67     67 EOCIA EQU 02H ;R/W - OCIA REGISTERS
68     68 EFCRST EQU 03H ;W - RESET FCA
69     69 EPRDEC EQU 04H ;R - DECODED PARITY BYTE
70     70 ;
71     71 ; PARITY DECODE BYTE
72     72 ;
73     73 EDECOO EQU 00001111B ;0-9 IS DECODED DIGIT
74     74 EDECSL EQU 00001010B ;A IS SL
75     75 EDECSR EQU 00001011B ;B IS BR

```

ISIS-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
 CHA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 2

5

10

15

20

25

30

35

40

45

50

55

LOC	OBJ	LINE	SOURCE STATEMENT
0000		= 77 EDECAR EQU 00001101B	;D IS AR
000E		= 78 EDECRE EQU 00001110B	;E IS NOT USED
000F		= 79 EDECRF EQU 00001111B	;F IS "NO DECODE" (ERROR)
0010		= 80 EDECE EQU 00010000B	;E-TAG
0020		= 81 EDECD EQU 00100000B	;D-TAG
0040		= 82 EDECBK EQU 01000000B	;BACKWARD CAPTURE
0080		= 83 EDECB7 EQU 10000000B	;NOT USED. ALWAYS=1.
		= 84 ;	
		= 85 ; SHIFT REGISTER READ	
		= 86 ;	
000F		= 87 ESRCHR EQU 00001111B	;BCD CHARACTER
0010		= 88 ESR4CN EQU 00010000B	;4-CHAR CAPTURE
0020		= 89 ESRF13 EQU 00100000B	;FRAME 1 OR 3 CAPTURE
0040		= 90 ESRPER EQU 01000000B	;PERIODICAL CAPTURE
0080		= 91 ESRSDT EQU 10000000B	;SDATA BYTE AVAILABLE
		= 92 ;	
		= 93 ; FLAG REGISTERS:	
		= 94 ;	
		= 95 ; R80-R4 SCAN FLAGS	
		= 96 ;	
0001		= 97 ESCNG EQU 00000001B	;SCANNING (FLAG CKFCA TO GET SEGMENTS)
0002		= 98 ER481 EQU 00000010B	;NOT USED
0004		= 99 ER482 EQU 00000100B	;NOT USED
0008		= 100 ESBFUL EQU 00001000B	;SEND BUFFER HAS DATA TO SEND
0010		= 101 ER484 EQU 00010000B	;NOT USED
0020		= 102 EBFREQ EQU 00100000B	;BUFMAN REQUEST FLAG
0040		= 103 ER486 EQU 01000000B	;NOT USED
0080		= 104 ER487 EQU 10000000B	;NOT USED
		= 105 ;	
		= 106 ; R80-R4 VERSION POINTER/FLAG	
		= 107 ;	
0000		= 108 EVER00 EQU 00H	;NO VALID VERSIONS
0001		= 109 EVERA EQU 01H	;UPC-A
0002		= 110 EVER13 EQU 02H	;EAN-13
0003		= 111 EVERE EQU 03H	;UPC-E
0004		= 112 EVER8 EQU 04H	;EAN-8
0005		= 113 EVERD1 EQU 05H	;UPC-D1
0006		= 114 EVERD2 EQU 06H	;UPC-D2
0007		= 115 EVERD3 EQU 07H	;UPC-D3
0008		= 116 EVERD4 EQU 08H	;UPC-D4
0009		= 117 EVERD5 EQU 09H	;UPC-D5
		= 118 ;	
0010		= 119 ER684 EQU 00010000B	;NOT USED
0020		= 120 ER685 EQU 00100000B	;NOT USED
0040		= 121 ER686 EQU 01000000B	;NOT USED
0080		= 122 ER687 EQU 10000000B	;NOT USED
		= 123 ;	
		= 124 ; SCANNER CONFIGURATION BYTE - CONFIG	
		= 125 ;	
0001		= 126 E6CN2S EQU 00000001B	;4 CHAR SEG 2 SCAN BIT
0002		= 127 E6CN2S EQU 00000010B	;6 CHAR SEG 2 SCAN BIT
		= 128 ;	
		= 129 ; TIMER CONSTANTS:	
		= 130 ;	
0004		= 131 E80MS EQU 4	;80 MSEC
000A		= 132 E200MS EQU 10	;200 MSEC
0018		= 133 E480MS EQU 24	;480 MSEC
0032		= 134 E1000M EQU 50	;1000 MSEC, 1.0 SECOND
0064		= 135 E2000M EQU 100	;2000 MSEC, 2.0 SECOND
		= 136 ;	
0004		= 137 ECDTON EQU 4	;80 MSEC, GOOD TONE ON TIME
0014		= 138 EBDTON EQU 20	;400 MSEC, BAD TONE ON TIME
		= 139 ;	
0028		= 140 ETONCT EQU 40	;TONE COUNT (CYCLES/20MS)
FFFA		= 141 ETONFQ EQU -6	;TONE FREQUENCY CONSTANT (500US)
		= 142 ;	
		= 143 ; SUPER-8 INTERFACE EXTERNAL MEMORY ADDRESS	
0008		= 144 ESUP8 EQU 00H	
		= 145 ;	
		= 146 ; COMMUNICATIONS ROUTINE CONSTANTS	
0088		= 147 EMSXBY EQU 08BH	;MISSCAN BYTE FOR SEND BUFFER
00CC		= 148 ETRMBY EQU 0CCB	;TERMINATION BYTE FOR SEND BUFFER
		= 149 ;	
		= 150 ; COMMUNICATIONS ROUTINE RECEIVE COMMANDS (SUPER-8 TO 8039)	
0011		= 151 ENSCAN EQU 11H	;ENABLE SCANNING (LASER ON)
0012		= 152 DISCAN EQU 12H	;DISABLE SCANNING (LASER OFF)

5 1515-11 MCS-68/UP1-41 MACRO ASSEMBLER, V4.2
 00AD95 ASSEMBLED 2/22/88 BY BLAKE ISAACS PAGE 3

LOC	OBJ	LINE	SOURCE STATEMENT
	0018	= 154 DIBEEP EQU 18H	
	0032	= 155 COMRST EQU 32H	;DISABLE TONE AFTER GOOD READ
	0077	= 156 IFRSHC EQU 77H	;RESET SCANNER COMMAND
10		= 157 ;	;1/F ROM CHECKSUM GOOD "COMMAND"
		= 158 ; FLATTOP TIME CONSTANTS	
	0008	= 159 EWAIT EQU 8	;VALUE FOR "NO SECS" WAIT
	002A	= 160 ECDLTV EQU 50-EWAIT	;CD-LT ON WAIT CONSTANT
	0016	= 161 EDRDLF EQU 30-EWAIT	;DOUBLE READ WAIT, IBM-OCR, 0.6 SEC
	002A	= 162 EDRDLF EQU 50-EWAIT	;DOUBLE READ WAIT, FUJITSU, 1.0 SEC
		= 163 ;	
15	0030	= 164 ; RAM POINTERS AND CONSTANTS USED IN THE MORSE TEST	
	0034	= 165 TSEG1 EQU 30H	;SEGMENT BUFFER
	0038	= 166 TSEG2 EQU 34H	
	003A	= 167 TCNT1 EQU 38H	;SEGMENT COUNTER
	0050	= 168 TCNT2 EQU 3AH	
	0001	= 169 TSCBUF EQU 50H	;FCA READ BUFFER
		= 170 ENCTRL EQU 001H	;MORSE CONTROL BYTE
20		= 171 ;	
		= 172 ; DATA MEMORY MAP - RAM - 8039 NEEDED	
		= 173 ;	
	0000	= 174 ; REGISTER BANK 0 (NON-INTERRUPT USAGE)	
	0000	= 175 ORG 000H	
	0004	= 176 MRB0: DS 4	;R0 TO R3 - SCRATCH
	0005	= 177 SCHFLG: DS 1	;R4 - SCAN FLAGS
	0006	= 178 DRTMR: DS 1	;R5 - DOUBLE READ TIMER
	0007	= 179 VERFLG: DS 1	;R6 - VERSION POINTER/FLAG
25		= 180 TIMREG: DS 1	;R7 - GENERAL PURPOSE TIMER/COUNTER
		= 181 ;	
	0008	= 182 ; STACK AREA	
		= 183 STACK: DS 16	;8 LEVELS OF SUBROUTINES ALLOWED
		= 184 ;	
	0018	= 185 ; REGISTER BANK 1 (INTERRUPT USAGE)	
	0018	= 186 MRB1: DS 3	;R0 TO R2 - SCRATCH (NOT USED)
30	001C	= 187 MRB1R3: DS 1	;R3 - GOOD READ TONE DISABLE FLAG
	001D	= 188 MRB1R4: DS 1	;R4 - NOT USED
	001E	= 189 TONCNT: DS 1	;R5 - TONE CYCLE COUNTER (CYCLES/20MS)
	001F	= 190 TONLTH: DS 1	;R6 - TONE LENGTH COUNTER
		= 191 TASAVE: DS 1	;R7 - TIMER "A" SAVE REGISTER
		= 192 ;	
		= 193 ; FREE MEMORY AREA	
		= 194 ;	
35	0020	= 195 ; SEGMENT BUFFERS	
	0020	= 196 SEGBUF EQU 8	
		= 197 SCHBUF: DS 4	;SCAN BUFFER
		= 198 ;	
	0024	= 199 BF6CST EQU 8	
	0024	= 200 L6S1: DS 4	;SCAN 1 BUFFER
	0028	= 201 L6S2: DS 4	;SCAN 2 BUFFER
	002C	= 202 L6SCNT: DS 1	;PACKED SCAN COUNTER (SCAN2/SCAN1)
40	002D	= 203 L6STOT: DS 1	;TOTAL COUNTER
		= 204 ;	
	002E	= 205 R6S1: DS 3	
	0031	= 206 R6S2: DS 3	
	0034	= 207 R6SCNT: DS 1	
	0035	= 208 R6STOT: DS 1	
	0012	= 209 BF6CNT EQU 8-BF6CST	
		= 210 ;	
45	0036	= 211 BF6CST EQU 8	
	0036	= 212 L4S1: DS 2	
	0038	= 213 L4S2: DS 2	
	003A	= 214 L4SCNT: DS 1	
	0038	= 215 L4STOT: DS 1	
		= 216 ;	
	003C	= 217 R4S1: DS 2	
	003E	= 218 R4S2: DS 2	
50	0040	= 219 R4SCNT: DS 1	
	0041	= 220 R4STOT: DS 1	
		= 221 ;	
	0042	= 222 N1S1: DS 2	
	0044	= 223 N1S2: DS 2	
	0046	= 224 N1SCNT: DS 1	
	0047	= 225 N1STOT: DS 1	
		= 226 ;	
55	0048	= 227 N2S1: DS 2	
	004A	= 228 N2S2: DS 2	
	004C	= 229 N2SCNT: DS 1	

ISIS-11 MCS-48/LP1-41 MACRO ASSEMBLER, V4.2
 QWAD95 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 4

5

10

15

20

25

30

35

40

45

50

55

```

LOC OBJ      LINE      SOURCE STATEMENT
      = 231 ;
004E      = 232 M3S1: DS      2
0050      = 233 M3S2: DS      2
0052      = 234 M3SCNT: DS     1
0053      = 235 M3STOT: DS     1
      = 236 ;
0054      = 237 M4S1: DS      2
0056      = 238 M4S2: DS      2
0058      = 239 M4SCNT: DS     1
0059      = 240 M4STOT: DS     1
      = 241 ;
005A      = 242 M5S1: DS      2
005C      = 243 M5S2: DS      2
005E      = 244 M5SCNT: DS     1
005F      = 245 M5STOT: DS     1
      = 246 ;
0060      = 247 M6S1: DS      2
0062      = 248 M6S2: DS      2
0064      = 249 M6SCNT: DS     1
0065      = 250 M6STOT: DS     1
0030      = 251 BF4CNT EQU     $-BF4CST
      = 252 ;
      = 253 ; SEND BUFFER
0066      = 254 SBFPNT: DS      1 ;POINTER
      = 255 ;
0067      = 256 SBUFAD EQU      $ ;FIRST DATA BYTE ADDRESS
0067      = 257 SBUFF: DS      16 ;DATA BUFFER
00CE      = 258 SBSTRT EQU     2*SBUFF ;PACKED BUFFER START POINTER
0012      = 259 SBUFFSZ EQU     $-SBUFF ;BYTES IN SEND BUFFER
0078      = 260 SBFEED EQU     $-1 ;LAST RAM LOCATION IN BUFFER
      = 261 ;
      = 262 ; WORK AREA USED BY ENOD10 ROUTINE
0079      = 263 WRKBUF: DS      3
      = 264 ;
      = 265 ; DOUBLE READ LABEL DATA SUM LOCATION
007C      = 266 DRSUM: DS      1
      = 267 ;
      = 268 ; SCANNER CONFIGURATION BYTE LOCATION
007D      = 269 CONFIG: DS     1
      = 270 ;
007D      = 271 LSTUSD EQU     $-1 ;LAST USED RAM LOCATION
      = 272 ; FVECTR.SRC INCLUDES FTIMER.SRC
      = 273 $ INCLUDE(:F1:FVECTR.SRC)
      = 274 ;-----
0000      = 275 ; FILE: FVECTR.SRC 06-16-86 13:00 BOB ACTIS
0000 E5    = 276 ;
0001 649F  = 277 ; RESET AND INTERRUPT VECTORS
      = 278 ;
      = 279 ORG      000H ;RESET TRAP
0000      = 280 RSTTRP: SEL     M80
0001 649F  = 281 JMP      P0MUP ;GO START PROGRAM
      = 282 ;
0003      = 283 ORG      003H ;EXTERNAL INTERRUPT TRAP
0003 93    = 284 INTTRP: RETR    ;RETURN FROM SPURIOUS INTERRUPTS
      = 285 ;
0007      = 286 ORG      007H ;INTERNAL TIMER INTERRUPT TRAP
0007      = 287 TIMTRP EQU     $ ;GO TO TIMER ROUTINE
      = 288 $ INCLUDE(:F1:FTIMER.SRC)
1= 289 ;-----
1= 290 ; FILE: FTIMER.SRC 10-08-86 15:40 BOB ACTIS
1= 291 ; FUNCTION: IF NO TONE IN PROGRESS, DECREMENT R80-R5 & R7 UNTIL 0.
1= 292 ; IF TONE IN PROGRESS, DECREMENT R81-R6 UNTIL 0.
1= 293 ; ENTRY: R81-R6 = TONE LENGTH IN 20'S OF MS.
1= 294 ; R81-R4 = IBM-4683 CLOCK TIMER
1= 295 ; R80-R5 = DOUBLE READ TIMER COUNTER
1= 296 ; R80-R7 = GENERAL PURPOSE TIMER COUNTER
1= 297 ; EXIT: R81-R7 = ACCUMULATOR SAVE
1= 298 ; R81-R5 = TONE CYCLE COUNTER
1= 299 ; R81-R6 = DECREMENTED UNTIL 0
1= 300 ; R81-R4 = DECREMENTED UNTIL 0
1= 301 ; R80-R5 = DECREMENTED UNTIL 0
1= 302 ; R80-R7 = DECREMENTED UNTIL 0
0007 05    1= 303 TIMER: SEL     R81
0008 AF    1= 304 MOV      R7,A ;SAVE A
0009 FE    1= 305 MOV      A,R6 ;GET TONE COUNTER
000A 961F  1= 306 JNZ      TIRE30 ;JUMP IF TONE IN PROGRESS

```

ISIS-II MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
GHA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 5

```

5      LOC OBJ      LINE      SOURCE STATEMENT
      000C FC      1= 308      MOV      A,R4
      000D C610     1= 309      JZ       TIME02      ;JUMP IF IBM-4683 TIMER = 0
      000F CC      1= 310      DEC      R4
      1= 311 ;
      0010 C5      1= 312 TIME02: SEL      R80
      0011 FF      1= 313      MOV      A,R7
      0012 C615     1= 314      JZ       TIME05      ;JUMP IF TIMER COUNTER = 0
      0014 CF      1= 315      DEC      R7
      1= 316 ;
      0015 FD      1= 317 TIME05: MOV      A,R5
      0016 C619     1= 318      JZ       TIME10      ;JUMP IF DOUBLE READ TIMER = 0
      0018 CD      1= 319      DEC      R5
      1= 320 ;
      0019 2306     1= 321 TIME10: MOV      A,#-250      ;20MS/80US = 250
      001B 62      1= 322 TIME20: MOV      T,A          ;SET TIMER COUNTER
      001C D5      1= 323      SEL      R81
      001D FF      1= 324      MOV      A,R7
      001E 93      1= 325      RETR
      1= 326 ;
      001F 09      1= 327 TIME30: IN      A,P1          ;GET TONE BIT
      0020 997F     1= 328      ANL      P1,#255-ETONE    ;SET TONE BIT LOW
      0022 F226     1= 329      JBT      TIME40          ;JUMP IF TONE BIT WAS HIGH
      0024 8980     1= 330      ORL      P1,#ETONE        ;SET TONE BIT HIGH
      1= 331 ;
      0026 55      1= 332 TIME40: STRT      T          ;CLEAR THE PRESCALAR
      0027 ED38     1= 333      DJNZ     R5,TIME60      ;JUMP IF NOT 20MS YET
      0029 BD28     1= 334      MOV      R5,#ETONCT      ;SET TONE CYCLE COUNTER
      1= 335 ;
      002B FC      1= 336      MOV      A,R4
      002C C62F     1= 337      JZ       TIME45      ;JUMP IF IBM-4683 TIMER = 0
      002E CC      1= 338      DEC      R4
      1= 339 ;
      002F C5      1= 340 TIME45: SEL      R80
      0030 FF      1= 341      MOV      A,R7
      0031 C634     1= 342      JZ       TIME50      ;JUMP IF TIMER COUNTER = 0
      0033 CF      1= 343      DEC      R7
      1= 344 ;
      0034 D5      1= 345 TIME50: SEL      R81
      0035 EE38     1= 346      DJNZ     R6,TIME60      ;JUMP IF TONE NOT FINISHED
      0037 8980     1= 347      ORL      P1,#ETONE        ;LEAVE TONE LINE HIGH
      0039 0419     1= 348      JMP      TIME10
      1= 349 ;
      003B 23FA     1= 350 TIME60: MOV      A,#ETONFO      ;SET TONE FREQ CONSTANT
      003D 0418     1= 351      JMP      TIME20
      352 $      INCLUDE(:F1:FSDATA.SRC)
      353 ;
      354 ; FILE: FSDATA.SRC 10-06-86 14:05 808 ACT15
      355 ;
      356 ; ROUTINE: SDATA
      357 ; FUNCTION: CLEAR THE SDATA BYTE THEN RETURN.
      358 ; THE 750SL DOES NOT IMPLEMENT MORSE OR RESET FROM SDATA
      359 ; ENTRY: R80
      360 ; SDATA READY IN FCA
      361 ; EXIT: USES R0,A
      362 ;
      003F 8802     = 363 SDATA: MOV      R0,#EOCIA      ;ENTERED FROM CKFCA ROUTINE
      0041 80      = 364      MOVX     A,R0          ;GET SDATA BYTE
      = 365 ;
      0042 D301     = 366 SDATA4: XRL      A,#ENCNTL
      0044 9656     = 367      JNZ      SDATA9          ;JUMP IF NOT THE MORSE CONTROL BYTE
      0046 27      = 368 SDATA8: CLR      A          ;CLEAR STACK POINTER/RETURN LINKAGE SINCE WE
      0047 D7      = 369      MOV      PSW,A          ;WILL JUMP INTO MORSE TEST AND STAY THERE
      0048 C5      = 370      SEL      R80          ;SETUP FOR MORSE TEST ENTRY
      0049 85      = 371      CLR      F0
      004A 95      = 372      CPL      F0          ;SET FLAG FOR CONTROL BYTE RECEIVED
      004B A5      = 373      GMR-     F1
      004C 8830     = 374      MOV      R0,#TSEG1
      004E 8810     = 375      MOV      R3,#16
      0050 99DF     = 376      ANL      P1,#255-ECOLT
      0052 8952     = 377      ORL      P1,#BOLT+ELASDB+EXTREB ;BOLT & MOTOR ON, LASER OFF
      0054 840A     = 378      JMP      TMOOS          ;BOLT INDICATES CNTL BYTE RECEIVED
      = 379 ;
      0056 83      = 380 SDATA9: RET
      381 ;-----;SPURIOUS SDATA
      0057 A3      382 TROPGD: MOVX     A,BA
      0058 83      383      RET
55

```

ISIS-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
GMA095 ASSEMBLED 2/22/88 BY BLAICE ISAACS

PAGE 6

```

5      LOC OBJ      LINE      SOURCE STATEMENT
      385 $          INCLUDE(=F1:TRONSH.SRC)
      386 ;
      387 ; ROUTINE: TRONSH 10-07-86 10:40 BOB ACTIS
      388 ;
10     389 ; FUNCTION - SUM ALL BYTES IN BOTH MEMORY BANKS
      390 ; ASSUMES THAT EACH OF THE SIXTEEN PAGES CONTAINS THE
      391 ; SEQUENCE "TROPGX: MOVP A,BA ; RET" FOR PAGE X.
      392 ;
      0100 B908      393 TRONSH: MOV R1,#STACK+3 ; R1 POINTS TO SECOND STACK ENTRY
      394 ;
      395 ; DO SIXTEEN PAGES WITH 256 BYTES PER PAGE
      396 ;
15     0102 27      397 CLR A
      0103 A1      398 MOV DR1,A ; STACK+3 = PAGE 0 TO START
      0104 AA      399 MOV R2,A ; R2 = BYTE ADDRESS
      0105 B810     400 MOV R3,#16 ; R3 = PAGES TO DO
      0107 A8      401 MOV R0,A ; R0 = SUM OF BYTES
      0108 D7      402 MOV PSU,A ; INSURE STACK IS EMPTY
      403 ;
      404 ; PUT PAGE ACCESS ADDRESS IN STACK
      405 ;
20     0109 F1      406 TRO10: MOV A,DR1
      010A D321     407 ADD A,#LOW TROTAB
      010C A3      408 MOVP A,BA
      010D C9      409 DEC R1
      010E A1      410 MOV DR1,A
      010F 19      411 INC R1
25     412 ;
      413 ; DO A PAGE
      414 ;
      0110 341C     415 TRO20: CALL TRO50 ; FETCH BYTE
      0112 68      416 ADD A,R0 ; ADD TO SUM
      0113 A8      417 MOV R0,A
      0114 EA10     418 DJNZ R2,TRO20 ; JMP = NOT DONE WITH PAGE
      0116 11      419 INC DR1 ; PAGE NUMBER INCREMENTED
30     0117 E809     420 DJNZ R3,TRO10 ; JMP = NOT THRU WITH PAGES
      0119 17      421 INC A ; (A) = ZERO FOR CORRECT SUM
      011A 644F     422 JMP TRORET
      423 ;
      424 ; LINK TO EACH PAGE
      425 ;
      011C 2302     426 TRO50: MOV A,R02
      011E D7      427 MOV PSU,A ; SET STACK POINTER AHEAD
      011F FA      428 MOV A,R2 ; A = ADDRESS OF BYTE TO FETCH
35     0120 83      429 RET ; JMP TO SELECTED PAGE
      430 ;
      431 ; TABLE FOR ADDRESS OF FETCH ROUTINE IN EACH PAGE
      432 ;
      0121 57      433 TROTAB: DB LOW TROPG0
      0122 76      434 DB LOW TROPG1
40     0123 80      435 DB LOW TROPG2
      0124 E3      436 DB LOW TROPG3
      0125 C6      437 DB LOW TROPG4
      0126 43      438 DB LOW TROPG5
      0127 D3      439 DB LOW TROPG6
      0128 6E      440 DB LOW TROPG7
      0129 E4      441 DB LOW TROPG8
45     012A D4      442 DB LOW TROPG9
      012B CC      443 DB LOW TROPGA
      012C D7      444 DB LOW TROPG8
      012D EB      445 DB LOW TROPGC
      012E E3      446 DB LOW TROPGD
      012F A8      447 DB LOW TROPGE
      0130 AD      448 DB LOW TROPGF
      449 $          INCLUDE(=F1:TRAN.SRC)
      450 ;
50     451 ; FILE: TRAN.SRC 6-19-86 11:20 BOB ACTIS
      452 ; FUNCTION: TEST THE 8039 RAM LOCATIONS 0 TO 7FH
      453 ; ENTRY: NO SETUP
      454 ; EXIT: RAM HAS GARBAGE (TEST PATTERN)
      455 ;
      456 ; START BY WRITTING EACH RAM ADDRESS INTO ITSELF
      457 TRAN: MOV R0,#7FH ; SIZE OF 8039 RAM
      0131 B87F     458 TRAN10: MOV A,R0 ; GET RAM ADDRESS
      0133 F8      459 MOV DR0,A ; STORE RAM ADDRESS IN IT'S LOCATION
65     0134 A0      459 MOV DR0,A ; STORE RAM ADDRESS IN IT'S LOCATION
      0135 E833     460 DJNZ R0,TRAN10 ; DO ALL LOCATIONS

```

5

 ISIS-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
 GHA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 7

LOC	OBJ	LINE	SOURCE STATEMENT
		= 462 ;	CHECK IF EACH LOCATION HAS IT'S OWN ADDRESS
	0137 B87F	= 463	MOV RO,87FH
	0139 F8	= 464	TRAN20: MOV A,RO ;GET RAM ADDRESS
10	013A D0	= 465	XRL A,DRO ;COMPARE ADDRESS TO CONTENTS
	0138 9674	= 466	JNZ TRAMER ;JUMP IF ERROR
	0130 E839	= 467	DJNZ RO,TRAN20 ;DO ALL LOCATIONS
		= 468 ;	
		= 469 ;	TRY A OSSH/OAAN CHECKER BOARD PATTERN
	013F B802	= 470	MOV RO,82 ;POINTER WILL GO FROM 2 TO 7FH
	0141 2355	= 471	TRAN30: MOV A,OSSH
15	0143 A0	= 472	MOV BRO,A ;WRITE EVEN LOCATION
	0144 37	= 473	CPL A ;PATTERN IS NOW OAAN
	0145 18	= 474	INC RO
	0146 A0	= 475	MOV BRO,A ;WRITE ODD LOCATION
	0147 18	= 476	INC RO
	0148 F8	= 477	MOV A,RO ;GET NEXT ADDRESS
	0149 37	= 478	CPL A
	014A F241	= 479	JBT TRAN30 ;JUMP IF NOT DONE YET, RO < 80H
		= 480 ;	
20		= 481 ;	CHECK THE OSSH/OAAN TEST PATTERN
	014C B902	= 482	MOV R1,82 ;USE R1 AS POINTER IN THIS SECTION
	014E F1	= 483	TRAN40: MOV A,BR1 ;GET EVEN BYTE DATA, OSSH
	014F 19	= 484	INC R1
	0150 61	= 485	ADD A,BR1 ;ADD ODD BYTE DATA, OAAN
	0151 17	= 486	INC A ;OSSH+OAAN+1=000H
	0152 9674	= 487	JNZ TRAMER ;JUMP IF ERROR
25	0154 19	= 488	INC R1
	0155 F9	= 489	MOV A,R1 ;GET NEXT ADDRESS
	0156 37	= 490	CPL A
	0157 F24E	= 491	JBT TRAN40 ;JUMP IF NOT DONE YET
		= 492 ;	
		= 493 ;	TRY A OAAN/OSSH CHECKER BOARD PATTERN
	0159 B902	= 494	MOV R1,82 ;POINTER WILL GO FROM 2 TO 7FH
	015B 23AA	= 495	TRAN50: MOV A,OAAH
30	0150 A1	= 496	MOV BR1,A ;WRITE EVEN LOCATION
	015E 37	= 497	CPL A ;PATTERN IS NOW OSSH
	015F 19	= 498	INC R1
	0160 A1	= 499	MOV BR1,A ;WRITE ODD LOCATION
	0161 19	= 500	INC R1
	0162 F9	= 501	MOV A,R1 ;GET NEXT ADDRESS
	0163 37	= 502	CPL A
35	0164 F258	= 503	JBT TRAN50 ;JUMP IF NOT DONE YET, R1 < 80H
		= 504 ;	
		= 505 ;	CHECK THE OAAN/OSSH TEST PATTERN
	0166 B802	= 506	MOV RO,82 ;USE RO AS POINTER IN THIS SECTION
	0168 F0	= 507	TRAN60: MOV A,DRO ;GET EVEN BYTE DATA, OAAN
	0169 18	= 508	INC RO
	016A 60	= 509	ADD A,DRO ;ADD ODD BYTE DATA, OSSH
	016B 17	= 510	INC A ;OAAN+OSSH+1=000H
40	016C 9674	= 511	JNZ TRAMER ;JUMP IF ERROR
	016E 18	= 512	INC RO
	016F F8	= 513	MOV A,RO ;GET NEXT ADDRESS
	0170 37	= 514	CPL A
	0171 F268	= 515	JBT TRAN60 ;JUMP IF NOT DONE YET
		= 516 ;	
	0173 27	= 517	CLR A ;A=0 INDICATES TEST PASSED
	0174 4457	= 518	TRAMER: JNP TRARET ;RETURN FROM RAM TEST
45		= 519 ;	INCLUDE(=F1:TOCIA.SRC) ; 750F ONLY <----
	0176 A3	= 520 ;
	0177 83	= 521	TROP61: MOVP A,BA
	0200	= 522	RET
		= 523	ORG 200H
		= 524 ;	INCLUDE(=F1:TTAG.SRC)
		= 525 ;
50		= 526 ;	ROUTINE: TTAG 06-19-86 15:30 BOB ACTIS
		= 527 ;
		= 528 ;	FUNCTION: CHECK DIGITAL LOGIC FOR CAPTURE OF 6 INCREASINGLY
		= 529 ;	LARGER TAGS - 012345 678912 - THE SYMBOL CAPTURE
		= 530 ;	PROCESSING IS USED TO COLLECT THE SEGMENTS FROM THE FCA.
		= 531 ;
	0200 FE	= 532	TTATAB: DB OFER,9AH,64H,16H,0B7H,000H,0A8H
	0201 9A	=	
	0202 64	=	
	0203 16	=	
55	0204 87	=	
	0205 D0	=	

ISIS-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
 CHA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 8

5

10

15

20

25

30

35

40

45

50

55

LOC	OBJ	LINE	SOURCE STATEMENT
0207	3A	= 533	DB 3AH,0B9H,42H,0B6H,59H,2EH,03EH
0208	89	=	
0209	42	=	
020A	86	=	
020B	59	=	
020C	2E	=	
020D	3E	=	
020E	8803	= 534	TTAG: MOV R0,#EFCRST
0210	90	= 535	MOVX DRO,A ;RESET FCA
0211	8901	= 536	ORL P1,#ETEST ;TEST CONTROL ACTIVE
0213	8800	= 537	MOV R0,#LOW TTATAB ; R0 POINTS TO CHAR GEN TABLE
0215	8E06	= 538	MOV R6,#06H ;THIS ROUTINE CHECKS DIGITAL LOGIC
0217	8F06	= 539	MOV R7,#06H ;FOR CAPTURE OF 6 INCREASINGLY LARGER
0219	8908	= 540	MOV R1,#08H ;TAGS 012345 678912.....
021B	880E	= 541	MOV R3,#0EH ;UNPACK DATA CONSTANT
021D	F8	= 542	UNPK1: MOV A,R0
021E	8A04	= 543	MOV R2,#04H ;UNPACK DATA CONSTANT
0220	A3	= 544	MOVX A,R4
0221	AC	= 545	MOV R4,A ;TEMP STORE
0222	5301	= 546	UNPK2: ANL A,#01H ;STRIP OUT ONE BIT
0224	E7	= 547	RL A
0225	17	= 548	INC A
0226	47	= 549	SWAP A
0227	AD	= 550	MOV R5,A
0228	FC	= 551	MOV A,R4
0229	77	= 552	RR A
022A	AC	= 553	MOV R4,A
022B	5301	= 554	ANL A,#01H
022D	E7	= 555	RL A
022E	17	= 556	INC A
022F	60	= 557	ADD A,R5
0230	A1	= 558	MOV DRI,A
0231	19	= 559	INC R1
0232	FC	= 560	MOV A,R4
0233	77	= 561	RR A
0234	AC	= 562	MOV R4,A
0235	EA22	= 563	DJNZ R2,UNPK2
0237	18	= 564	INC R0
0238	EB1D	= 565	DJNZ R3,UNPK1
		= 566 ;	
		= 567 ;	SETUP TEST BIT ON PORT 1
		= 568 ;	
023A	2390	= 569	MOV A,#ETEST+EP12+EP13+ELASDB+ETONE
023C	39	= 570	OUTL P1,A
		= 571 ;	
023D	8802	= 572	MOV R3,#02H
023F	8006	= 573	MOV R5,#06H
0241	8A38	= 574	TEST1: MOV R2,#38H ;START TAG POINTER
0243	883F	= 575	MOV R0,#3FH
0245	FD	= 576	TEST2: MOV A,R5 ;START TAG OUTPUT
0246	68	= 577	ADD A,R3
0247	AC	= 578	MOV R4,A
0248	EC48	= 579	TEST3: DJNZ R4,TEST3
024A	FD	= 580	MOV A,DRO
		= 581 ;	
024B	3251	= 582	JB1 TST31 ;JMP = SEND SPACE
024D	99FD	= 583	ANL P1,#OFFH-ETHARK
024F	4455	= 584	JMP TST32
0251	8902	= 585	TST31: ORL P1,#ETMARK
0253	00	= 586	NOP
0254	00	= 587	NOP
		= 588 ;	
0255	FD	= 589	TST32: MOV A,R5
0256	68	= 590	ADD A,R3
0257	AC	= 591	MOV R4,A
0258	ECS8	= 592	TEST4: DJNZ R4,TEST4
025A	FD	= 593	MOV A,DRO
		= 594 ;	
025B	47	= 595	SWAP A
025C	3262	= 596	JB1 TST41 ;JMP = SEND SPACE
025E	99FD	= 597	ANL P1,#OFFH-ETHARK
0260	4466	= 598	JMP TST42
0262	8902	= 599	TST41: ORL P1,#ETMARK
0264	00	= 600	NOP
0265	00	= 601	NOP
		= 602 ;	

5 1515-II MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
CHAD95 ASSEMBLED 2/22/88 BY BLAKE ISAACS PAGE 9

LOC	OBJ	LINE	SOURCE STATEMENT	
0267	EA45	= 604	DJNZ R2,TEST2	;END OF TAG
0269	ED41	= 605	DJNZ R5,TEST1	;END OF ALL TAGS
		= 606 ;		
0268	F5	= 607	SEL M81	
026C	1400	= 608	CALL CLR6SG	
026E	E5	= 609	SEL M80	;CLEAR 6 CHAR SEG BUFFS & COUNTERS
		= 610 ;		
026F	BE18	= 611	MOV R6,#24	;SHOULD BE ONLY 12 SEGS, TRY FOR MORE
0271	FC	= 612	MOV A,R4	
0272	4301	= 613	ORL A,#ESCHG	;SET SCAN FLAG SO CKFCA WILL
0274	AC	= 614	MOV R4,A	;PUT SEGMENTS INTO THE SCAN BUFF
0275	F5	= 615	GETLUP: SEL M81	
0276	14A4	= 616	CALL CKFCA	;GET SEG, IF ANY, FROM FCA
0278	5404	= 617	CALL PROCSG	;PUT SEG, IF ANY, INTO SEG BUFF
027A	E5	= 618	SEL M80	
027B	EET5	= 619	DJNZ R6,GETLUP	;GO CHECK FOR MORE SEGMENTS
		= 620 ;		
027D	85	= 621	CLR R0	
027E	B82C	= 622	MOV R0,#16SCNT	;SETUP FOR RIGHT HALF LOOP
0280	F0	= 623	TCCNT: MOV A,R0	
0281	18	= 624	INC R0	;GET L OR R COUNT
0282	60	= 625	ADD A,R0	
0283	03F4	= 626	ADD A,#-12	;GET L OR R TOTAL
0285	96A7	= 627	JNZ TTA90	;JUMP IF X6SCNT+X6STOT<12
		= 628 ;		
0287	B834	= 629	MOV R0,#R6SCNT	
0289	95	= 630	CPL R0	
028A	B680	= 631	JFO TCCNT	;JUMP TO DO RIGHT HALF
		= 632 ;		
028C	BA04	= 633	MOV R2,#4	;4 BYTES TO COMPARE
028E	B824	= 634	MOV R0,#16S1	
0290	B9A9	= 635	MOV R1,#LOW TTACHK	;DATA CHECK TABLE
0292	F9	= 636	LOOPCK: MOV A,R1	
0293	A3	= 637	MOV A,R2	
0294	D0	= 638	XRL A,R0	
0295	96A7	= 639	JNZ TTA90	;JMP IF BAD CHECK OF DATA
		= 640 ;		
0297	18	= 641	INC R0	
0298	19	= 642	INC R1	
0299	EA92	= 643	DJNZ R2,LOOPCK	
		= 644 ;		
029B	B82E	= 645	MOV R0,#R6S1	
029D	BA03	= 646	MOV R2,#3	;3 BYTES TO COMPARE
029F	95	= 647	CPL R0	
02A0	B692	= 648	JFO LOOPCK	;JUMP TO DO RIGHT HALF
		= 649 ;		
02A2	B87F	= 650	MOV R0,#7FH	
02A4	A0	= 651	CLR R0: MOV R0,A	;LOOP COUNTER, RAM SIZE
02A5	E8A4	= 652	DJNZ R0,CLR R0	;A=0 AT THIS POINT, TEST PASSED
		= 653 ;		
02A7	6469	= 654	TTA90: JMP TTARET	;A=0 FOR SUCCESSFUL COMPLETION
		= 655 ;		
02A9	01	= 656	TTACHK: DB 01H,23H,45H,0CH	
02AA	23	=		
02AB	45	=		
02AC	0C	=		
02AD	67	= 657	DB 67H,89H,12H	
02AE	89	=		
02AF	12	=		
		=		
02B0	A3	= 658 ;		
02B1	B3	= 659	TROP2: MOV A,R2	
0300		= 660	RET	
		= 661	ORG 300H	
		= 662 ;	INCLUDE(:F1:TMOTOR.SRC)	; NOT USED IN 750F <---
		= 663 ;		
		= 664 ;	FILE: TMOTOR.SRC 9-11-86 08:45	BOB ACTIS
		= 665 ;	FUNCTION: TEST THE MOTOR AND UP2SPD SIGNAL	
		= 666 ;	ENTRY: NO SETUP	
		= 667 ;	EXIT: USES R3,R7	
		= 668 ;		
0300	B912	= 669	TMOTOR: ORL P1,#ELASDB+ENTREB	;LASER OFF, MOTOR ON
0302	55	= 670	STRT T	
0303	25	= 671	EN TONT1	;ENABLE THE TIMER
		= 672 ;		
0304	BFB5	= 673	MOV R7,#5	
0306	FF	= 674	TMOT10: MOV A,R7	;SET TIMER FOR 100MSEC

1515-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
 QMAD95 ASSEMBLED 2/22/88 BY BLAICE ISAACS

PAGE 12

```

5      LOC OBJ      LINE      SOURCE STATEMENT

      = 826 ;
      = 827 ; SETUP PORTS
03A5 238F      = 828      MOV      A,#255-EVLSIR ;PWR RST THE VLSI (NO EFFECT ON 750F)
03A7 3A        = 829      OUTL     P2,A
03A8 8A40      = 830      ORL      P2,#EVLSIR ;ENABLE VLSI TO RUN
10 03AA 239C    = 831      MOV      A,#EP12+EP13+ETONE+ELASDB
03AC 39        = 832      OUTL     P1,A
      = 833 ;
      = 834 ; RESET THE FRAME CONTROL ARRAY
03AD 8803      = 835      MOV      R0,#EFCRST
03AF 90        = 836      MOVX     @R0,A
      = 837 ;
      = 838 ; GO PERFORM THE POWER UP TESTS.
      = 839 ; STICK IN TPON LOOP IF ANY FAILURES.
15 0380 6440    = 840      JMP      TPON
0382           = 841      TPORET   EQU      $
      = 842 ;
      = 843 ; TPON PASSED. START THE INTERNAL TIMER
0382 05        = 844      SEL      R81
0383 8028      = 845      MOV      R5,#ETONCT ;SET THE TONE CYCLE COUNTER
0385 05        = 846      SEL      R80
20 0386 55      = 847      STRT     T
0387 25        = 848      EN       TMTI
      = 849 ;
      = 850 ; GREEN LIGHT, GOOD POWER UP TONE, LASER ON AND WAIT 1 SEC.
0388 8920      = 851      ORL      P1,#ELDLT
038A 05        = 852      SEL      R81
038B 8E04      = 853      MOV      R6,#EBOMS ;200SEC BEEP FOR 750SL, 80MS FOR 750F
25 038D 05      = 854      SEL      R80
038E 99EF      = 855      ANL      P1,#255-ELASDB ;LASER ON, TIME TO START BEFORE RDTAG
03C0 8F32      = 856      MOV      R7,#E1000H
03C2 FF        = 857      TPON15: MOV      A,R7
03C3 96C2      = 858      JNZ      TPON15 ;WAIT
      = 859 ;
      = 860 ; REINITIALIZE AFTER TEST SEQUENCE
03C5 99CF      = 861      MOVZ0: ANL      P1,#255-(EGDLT+ELASDB) ;GOOD LIGHT OFF & LASER ON
03C7 8942      = 862      ORL      P1,#EBDLT+ENTREB ;BAD LIGHT ON & MOTOR ON (NO MOTOR ON F)
      = 863 ;
03C9 8803      = 864      MOV      R0,#EFCRST
03CB 90        = 865      MOVX     @R0,A ;RESET THE FCA
03CC 8F02      = 866      MOV      R7,#2 ;SET TIMER FOR 40 MSEC
03CE FF        = 867      MOVZ5: MOV      A,R7
03CF 96CE      = 868      JNZ      MOVZ5 ;WAIT FOR FCA TO SEE SEGS IF ANY
      = 869 ;
35 0301 27      = 870 ; CLEAR DATA MEMORY AND PSU
      = 871      CLR      A
0302 07        = 872      MOV      PSW,A ;CLEAR THE PSW
0303 887F      = 873      MOV      R0,#7FH
0305 A0        = 874      MOVZ0: MOV      @R0,A ;CLEAR MEMORY
0306 E8D5      = 875      DJNZ     R0,MOVZ0
      = 876 ;
40 0308 05      = 877      SEL      R81
0309 8028      = 878      MOV      R5,#ETONCT ;SET THE TONE CYCLE COUNTER
030B 05        = 879      SEL      R80
      = 880 ;
030C 887D      = 881      MOV      R0,#CONFIC
030E 8000      = 882      MOV      @R0,#0 ;SETUP CONFIGURATION REGISTER
      = 883 ;
45 03E0 F5      = 884      SEL      R81
03E1 C400      = 885      JMP      RDTAG
      = 886 ;
03E3 A3        = 887      TROP3: MOVP     A,#A
03E4 83        = 888      RET
04D0           = 889      ORG      400H
      = 890 ; INCLUDE(:"F1:NORSE1.SRC")
      = 891 ;
50 0392 : FILE: NORSE1.SRC NORSE TEST PART 1 OF 3.
      = 893 ; 07-03-86 09:05 BOB ACTIS
      = 894 ;
      = 895 ; ROUTINE: TNORSE
      = 896 ;
      = 897 ; NORSE TEST (MOST ORIGINATED SEGMENT EVALUATION) WILL RECEIVE
      = 898 ; TEST TAG DEFINITION FROM MOST, COUNT THE NUMBER OF TIMES IT
      = 899 ; "SEES" EACH SEGMENT OF THE TAG AND THEN SEND THE COUNT
      = 900 ; INFORMATION TO THE MOST....REMAINS IN NORSE TEST UNTIL
55 0391 : DSATA=0, OR POWER RESET.
      = 901 ;

```

```

5      1515-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2      PAGE 10
      0309 0A      = 676 ;
      030A 37      = 677      IN      A,P2
      030B 822F    = 678      CPL      A
10      = 679      JBS      TNOT50      ;JUMP IF NOT UP2SPD
      = 680 ;
      = 681 ; MOTOR OFF AND WAIT FOR IT TO SLOW. FLASH BOTH LIGHTS.
      030D 99FD    = 682      ANL      P1,#255-ENTREB ;MOTOR OFF
      030F 8432    = 683 TNOT20: MOV      R3,#50      ;SET LOOP COUNTER
      0311 8F05    = 684 TNOT22: MOV      R7,#5      ;SET TIMER FOR 100MSEC
      0313          = 685 TNOT24 EQU      $
      = 686 ;
      0313 FF      = 687      CALL     CX4HRS      ;CHECK FOR HORSE CONTROL BYTE (NOT IMPLEMENTED)
15      0314 9613  = 688      MOV      A,R7      ;20MSEC*5=50=5SEC
      = 689 ;
      0316 CB      = 690      DEC      R3      ;DECREMENT LOOP COUNTER
      0317 FB      = 691      MOV      A,R3
      0318 C623    = 692      JZ       TNOT28      ;JUMP IF FINISHED WAITING
      = 693 ;
      031A 09      = 694      IN      A,P1
20      031B 999F  = 695      ANL      P1,#255-(EGDLT+EBDLT) ;LIGHTS OFF
      031D 8211    = 696      JBS      TNOT22      ;JUMP IF GDLT WAS ON
      = 697 ;
      031F 8960    = 698      ORL      P1,#EGDLT+EBDLT ;LIGHTS ON
      0321 6411    = 699      JMP      TNOT22
      = 700 ;
      0323 999F    = 701 TNOT28: ANL      P1,#255-(EGDLT+EBDLT) ;LIGHTS OFF
      = 702 ;
25      0325 8902  = 703      ORL      P1,#ENTREB      ;MOTOR ON
      0327 8F05    = 704      MOV      R7,#5      ;SET TIMER FOR 100 MSEC
      0329 FF      = 705 TNOT40: MOV      A,R7
      032A 9629    = 706      JNZ      TNOT40      ;WAIT FOR MOTOR CIRCUIT TO POWER UP
      = 707 ;
      032C 0A      = 708      IN      A,P2
      032D 8246    = 709      JBS      TNOT90      ;JUMP IF ALREADY UP2SPD ... FAILED
      = 710 ;
30      = 711 ; WAIT 30 SECONDS FOR THE MOTOR TO GET UP2SPD. FLASH GREEN LIGHT.
      032F 8BFA    = 712 TNOT50: MOV      R3,#250      ;SET LOOP COUNTER
      0331 8F06    = 713 TNOT60: MOV      R7,#6      ;SET TIMER FOR 120 MSEC
      0333          = 714 TNOT80 EQU      $
      = 715 ;
      0333 FF      = 716      CALL     CX4HRS      ;CHECK FOR HORSE CONTROL BYTE (NOT IMPLEMENTED)
      0334 9633    = 717      MOV      A,R7      ;20MS*6*250=30SEC
      = 718 ;
      = 719 ;
35      0336 0A      = 719      IN      A,P2
      0337 8248    = 720      JBS      TNOT95      ;JUMP IF MOTOR IS UP2SPD ... PASSED
      = 721 ;
      0339 CB      = 722      DEC      R3      ;DECREMENT LOOP COUNTER
      033A FB      = 723      MOV      A,R3
      033B C646    = 724      JZ       TNOT90      ;JUMP IF TIMED OUT ... FAILED
      = 725 ;
40      033D 09      = 726      IN      A,P1
      033E 99DF    = 727      ANL      P1,#255-EGDLT ;TOGGLE GDLT WHILE WAITING FOR UP2SPD
      0340 8231    = 728      JBS      TNOT60      ;GDLT OFF
      = 729 ;
      = 730 ;
      0342 8920    = 730      ORL      P1,#EGDLT      ;GDLT ON
      0344 6431    = 731      JMP      TNOT60      ;CONTINUE WAITING FOR UP2SPD
      = 732 ;
45      = 733 ; COME HERE IF THE TEST FAILED
      0346 99FD    = 734 TNOT90: ANL      P1,#255-ENTREB ;MOTOR OFF
      0348 27      = 735      CLR      A
      0349 37      = 736      CPL      A
      034A 83      = 737      RET      ;SET FAILED FLAG
      = 738 ;
      = 739 ; COME HERE IS THE TEST PASSED
      034B 27      = 740 TNOT95: CLR      A      ;SET PASSED FLAG
      034C 83      = 741      RET
50      = 742 $      INCLUDE(=F1:TPONSL.SRC)
      = 743 ;
      = 744 ; FILE: TPONSL.SRC 4-16-87 DREW TAUSSIG
      = 745 ; FUNCTION: PERFORM SELF-TESTS FOR 750SL
      = 746 ;
      034D 2400    = 747 TPON: JMP      TROMSH
      034F C655    = 748 TRORET: JZ      TPOH20      ;JUMP IF THE CHECKSUM PASSED
      0351 8000    = 749      MOV      R0,R0      ;NO BEEPS WITH THIS ERROR
55      0353 647B  = 750      JMP      TPOH90
      = 751 ;

```

1515-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
GM4095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 11

```

5      LOC OBJ      LIKE      SOURCE STATEMENT

      0357 C650      = 753 TRARET: JZ      TPON30      ;JUMP IF THE RAM TEST PASSED
      0359 B801      = 754      MOV      R0,#1      ;ONE BEEP FOR THIS ERROR
      035B 647B      = 755      JMP      TPON90
      = 756 ;
10     035D F5       = 757 TPON30: SEL      R81
      035E 3400      = 758      CALL     RCONM      ;GET CHECKSUM BYTE FROM SUPERS
      0360 E5        = 759      SEL      R80
      0361 C667      = 760      JZ      TPON40      ;VALID CHCSUM SETS A TO 0
      0363 B805      = 761      MOV      R0,#5      ;5 BEEPS FOR 1/F ROM CHECKSUM ERROR
      0365 647B      = 762      JMP      TPON90
      = 763 ;
      0367 440E      = 764 TPON40: JMP      TTAG
15     0369 99FC      = 765 TTARET: ANL     P1,#255-(ETEST+ETMARK) ;CLEAR TEST AND TEST MARK LINES
      036B C671      = 766      JZ      TPON50      ;JUMP IF THE TTAG TEST PASSED
      036D 8803      = 767      MOV      R0,#3      ;THREE BEEPS FOR THIS ERROR
      036F 647B      = 768      JMP      TPON90
      = 769 ;
      0371 7400      = 770 TPON50: CALL     TMOTOR
      0373 C679      = 771      JZ      TPON60      ;JUMP IF THE MOTOR TEST PASSED
      0375 B804      = 772      MOV      R0,#4      ;FOUR BEEPS FOR THIS ERROR
20     0377 647B      = 773      JMP      TPON90
      = 774 ;
      0379 6482      = 775 TPON60: JMP      TPORET      ;RETURN FROM THE POWER UP TESTS
      = 776 ;
      037B 747F      = 777 TPON90: CALL     TERRUT      ;ERRORS COME HERE
      037D 0400      = 778      JMP      RSTTRP      ;RESTART THE PROGRAM
      = 779 ;
25     = 780 ; ROUTINE: TERRUT 6-17-86 16:25 BOB ACTIS
      = 781 ; FUNCTION: SELFTEST ERROR ROUTINE
      = 782 ; BEEP R0 TIMES AND WAIT 1 SECOND
      = 783 ; ENTRY: R0 = NUMBER OF BEEPS
      = 784 ; EXIT: USES R0, R7
      = 785 ;
      037F 990F      = 786 TERRUT: ANL     P1,#255-EGDLT ;GDLT OFF
      0381 8940      = 787      ORL     P1,#EBDLT ;BDLT ON
      = 788 ;
30     0383 55       = 789      STRT     T
      0384 25       = 790      EN      TCNT1      ;ENABLE THE TIMER OPERATION
      = 791 ;
      0385 F8       = 792 TERR02: MOV     A,R0
      0386 C69B      = 793      JZ      TERR10      ;JUMP IF NO BEEPS
      = 794 ;
      0388 D5       = 795 TERR04: SEL     R81
35     0389 B028      = 796      MOV     R5,#ETONCT ;SET THE TONE CYCLE COUNTER
      038B BE02      = 797      MOV     R6,#2      ;SET BEEP TIMER FOR 40 MSEC
      038D FE       = 798 TERR06: MOV     A,R6
      038E 968D      = 799      JNZ     TERR06      ;WAIT FOR BEEP TO END
      0390 C5       = 800      SEL     R80
      = 801 ;
      0391 BF03      = 802      MOV     R7,#3      ;SET TIMER FOR 60 MSEC
      0393 FF       = 803 TERR08: MOV     A,R7
40     0394 9693      = 804      JNZ     TERR08      ;WAIT BETWEEN BEEPS
      = 805 ;
      0396 E88B      = 806      DJNZ     R0,TERR04 ;BEEP LOOP
      = 807 ;
      0398 BF32      = 808 TERR10: MOV     R7,#E1000M ;SET TIMER FOR 1 SECOND
      039A FF       = 809 TERR12: MOV     A,R7
      039B 969A      = 810      JNZ     TERR12
      = 811 ;
45     039D 83       = 812      RET
      = 813 ; INCLUDE(:F1:FPOLUP.SRC)
      ;
      TROP63: MOV    A,2A
      RET
      ORG      400H
50     $          INCLUDE(:F1:NO          = 814 ;
      = 815 ; FILE: FPOLUP.SRC 12-09-86 13:50 BOB ACTIS
      = 816 ; ROUTINE: POLUP
      = 817 ; FUNCTION: INITIALIZE SYSTEM
      = 818 ;
      039E 93       = 819 POL00: RETR     $          ;RESET THE IIP FLIP-FLOP
      039F          = 820 POLUP  EQU     I
      039F 15       = 821      DIS     I
      03A0 35       = 822      DIS     TCNT1
      03A1 27       = 823      CLR     A
65     03A2 D7       = 824      MOV     PSW,A

```

```

5      LOC OBJ      LINE      SOURCE STATEMENT
      = 903 ; BYTE DEFINITION
      = 904 ;
      = 905 ; TEST CONTROL BYTE = 01H
      = 906 ; THDR = COUNT HEADER BYTE = 15H (AFTER PARITY INSERT = 95H)
      = 907 ; TTRLR = COUNT TRAILER BYTE = 2AH (AFTER PARITY INSERT = 6AH)
      = 908 ;
10     0400 85      = 909 TMO00: CLR F0 ;F0 INDICATES CONTROL BYTE RECEIVED IF ON
      0401 A5      = 910 CLR F1 ;F1 ON INDICATES ODD BYTE RECEIVED
      0402 8830     = 911 MOV R0,RTSEG1 ;R0=SEG TABLE POINTER
      0404 8810     = 912 MOV R3,#16 ;R3=LOOP COUNTER TO RECEIVE 16 DATA CHAR'S
      0406 999F     = 913 ANL P1,#255-(EGDLY+EBDLY) ;LIGHTS OFF
      0408 8910     = 914 ORL P1,RELASDR ;LASER OFF
      040A 860E     = 915 TMO05: JNZ TMO10 ;JMP IF FCA HAS DATA
      040C 840A     = 916 JMP TMO05
15     = 917 ;
      = 918 ;
      = 919 ;
      = 920 ;
      = 921 TMO10: MOV R1,#01H
      040E 8901     = 922 MOVX A,DR1 ;READ FCA S.R.
      0410 81      = 923 MOVX DR1,A ;DO FCA FRAME RESET
      0411 91      = 924 CPL A
      0412 37      = 925 J87 TMO05 ;RETURN IF FCA DOES NOT HAVE COMM DATA
      0413 F20A     = 926 ;
      = 927 ;
      = 928 ;
      = 929 INC R1 ;TO 02H TO READ COMM REG
      0415 19      = 930 MOVX A,DR1 ;READ FCA COMM REG
      0416 81      = 931 MOV R1,A ;R1=COMM DATA
      0417 A9      = 932 JNZ TMO14 ;JMP IF DATA NOT=0
      0418 961C     = 933 TMO12: JMP RSTRP ;JMP TO POWER UP RESET IF COMM DATA=0
      041A 0400     = 934 ;
      = 935 TMO14: CALL CPARTY ;CPARTY WILL CHK FOR CORRECT PARITY
      041C 840E     = 936 INC A
      041E 17      = 937 JNZ TMO00 ;JMP BACK TO START IF INCORRECT PARITY
      041F 9600     = 938 MOV A,R1 ;R1=COMM DATA
      0421 F9      = 939 JF0 TMO15 ;JMP IF CONTROL BYTE ALREADY RECEIVED
      0422 8620     = 940 ;
      = 941 ;
      = 942 ;
      0424 0301     = 941 XRL A,#001H
      0426 9600     = 942 JNZ TMO00 ;JMP BACK TO START IF NOT CONTROL BYTE
      0428 95      = 943 CPL F0 ;SET F0=1 TO INDICATE CONTROL BYTE RCVD
      0429 8940     = 944 ORL P1,REBOLY ;BO-1Y ON SAYS CTRL BYTE RCVD
      042B 840A     = 945 JMP TMO05 ;BACK TO TMO05 TO GET 16 DATA BYTES
      = 946 ;
      = 947 ;
      = 948 ;
35     = 949 TMO15: MOV A,R1 ;R1=COMM DATA
      042D F9      = 950 JF1 TMO20 ;JMP IF THIS IS ODD BYTE
      042E 7634     = 951 SHAP A
      0430 47      = 952 MOV DR0,A ;PUT EVEN BYTE IN HIGH NIBBLE
      0431 A0      = 953 JMP TMO29
      0432 8436     = 954 TMO20: XCHD A,DR0 ;PUT ODD BYTE IN LOW NIBBLE
      0434 30      = 955 INC R0 ;INC R0 TO NEXT SEG TABLE ADDRESS
      0435 18      = 956 ;
      = 957 ;
      = 958 ;
      = 959 TMO29: CPL F1
      0436 85      = 960 MOV A,R1
      0437 F9      = 961 J84 TMO00 ;JMP = CONTROL BYTE
      0438 9200     = 962 J85 TMO30 ;JMP = LAST BYTE RCVD
      043A 8240     = 963 DJNZ R3,TMO05 ;JMP = MORE TO COME
      043C E80A     = 964 JMP TMO00
      043E 8400     = 965 ;
      = 966 ; IF 16 BYTES RECEIVED, PERFORM TEST
      = 967 ;
      = 968 TMO30: DJNZ R3,TMO00 ;JMP = NOT 16 BYTES
      0440 E800     = 969 CALL TSCNT ;DO COLLECT SEGMENTS
      0442 9462     = 970 ;
      = 971 ; SEND HEADER, COUNTS AND LAST BYTE TO COMPUTER
      = 972 ;
      = 973 ;
      0444 8915     = 973 MOV R1,#15H ;SEND COUNT HEADER BYTE TO MOST
      0446 8400     = 974 CALL TNSHD ;SEND HEADER
      = 975 ;
      = 976 ;
      0448 8808     = 976 MOV R3,#8
      044A 8838     = 977 MOV R0,#TCTR1 ;R0=COUNT PONTER
      044C F8      = 978 TMO50: MOV A,R3
55

```

1S1S-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
GHA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 14

5

10

15

20

25

30

35

40

45

50

55

```

LOC OBJ      LINE      SOURCE STATEMENT
044F F0      = 980      MOV      A,DR0
0450 47      = 981      SWAP     A
0451 8455    = 982      JMP      TMO59
0453 30      = 983 TMO55: XCHD     A,DR0
0454 18      = 984      INC      R0
0455 530F    = 985 TMO59: ANL      A,#0FFH
0457 A9      = 986      MOV      R1,A
0458 B400    = 987      CALL     TMSHD
045A E84C    = 988      DJNZ     R3,TMO50
              = 989 ;
045C B92A    = 990      MOV      R1,#2AH      ;2AH = COUNT TRAILER BYTE
045E B400    = 991      CALL     TMSHD      ;SEND TRAILER BYTE
0460 B400    = 992      JMP      TMO00      ;FINISHED WITH THIS REQUEST, START OVER.
              = 993 ;
              = 994 ; INCLUDE(:F1:NORSE2.SRC)
              = 995 ; FILE: NORSE2.SRC NORSE TEST PART 2 OF 3.
              = 996 ; 07-03-86 10:35 BOB ACT15
              = 997 ;
              = 998 ; ROUTINE: TSCNT - COUNT TEST SEGMENTS
              = 999 ;
              =1000 ; FUNCTION: COUNT THE NUMBER OF TIMES EACH OF TWO SEGMENTS ARE DETECTED
              =1001 ; OVER A ONE SECOND PERIOD. TURN THE LASER ON DURING THE TEST.
              =1002 ; EXIT TO POWER ON ENTRY IF ANYTHING RECEIVED FROM MOST.
              =1003 ; WILL COMPARE 1ST 2 CHAR** OF SEGMENT CAPTURED WITH
              =1004 ; TSEG1 & TSEG2 AND RESET FCA SHIFT REG IF NO COMPARE.
              =1005 ; 4 CHAR SEC'S MUST BE PRECEDED BY 00H IN THE SEC TABLE...
              =1006 ;
              =1007 ; ENTRY:
              =1008 ; TSEG1 = SEGMENT 1 DEFINITION (4 BYTES).
              =1009 ; TSEG2 = SEGMENT 2 DEFINITION (4 BYTES)
              =1010 ;
              =1011 ; EXIT:
              =1012 ; TCHT1 = SEGMENT 1 COUNT (2 BYTES)
              =1013 ; TCHT2 = SEGMENT 2 COUNT (2 BYTES)
              =1014 ;
              =1015 ; *(A)
              =1016 ; *(R0)
              =1017 ; *(R1)
              =1018 ; *(TSCBUF) TO (TSCBUF+3)
              =1019 ; R80 (R7) = SECOND TIMER = 0
              =1020 ;
              =1021 ; PARAMETERS:
              =1022 ;
              =1023 ; SEGMENT DEFINITION TABLE
              =1024 ;
              =1025 ; BYTE 0 - 1ST AND 2ND CHAR (AS DETECTED OR ZERO IF 4-CHAR)
              =1026 ; BYTE 1 - 3RD AND 4TH CHAR (1ST & 2ND ON 4-CHAR)
              =1027 ; BYTE 2 - 5TH AND 6TH CHAR (3RD & 4TH ON 4-CHAR)
              =1028 ; BYTE 3 - DECODED PARITY WORD
              =1029 ;
0462 99EF    =1030 TSCNT: ANL      P1,#0FFH-ELASDB ;LASER ON
0464 8F02    =1031      MOV      R7,#02H
0466 B438    =1032      CALL     TMWAIT      ;WAIT FOR LASER TO TURN ON
0468 8F32    =1033      MOV      R7,#E1000H
046A 27      =1034      CLR      A
0468 B438    =1035      MOV      R0,#TCHT1
046D B909    =1036      MOV      R1,#9
046F A0      =1037 TSC02: MOV      DR0,A      ;CLEAR TCHT1 TO TCHT1+9
0470 18      =1038      INC      R0
0471 E96F    =1039      DJNZ     R1,TSC02
              =1040 ;
              =1041 ; RESET FCA TO CLEAR ANY SEGMENTS
              =1042 ;
0473 8803    =1043      MOV      R0,#EFCRST
0475 90      =1044      MOVX     DR0,A
              =1045 ;
              =1046 ; IF SYNCAP THEN READ THE SEGMENT; EXIT IF MOST BYTE RECEIVED.
              =1047 ;
0476 A5      =1048 TSC04: CLR      F1      ;F1 REMEMBERS TO INC TCHT1 OR TCHT2
0477 8801    =1049      MOV      R0,#ESR0
0479 868D    =1050 TSC05: JHI      TSC08      ;JMP IF SYNCAP OR MOST COMM. OCCURED
047B FF      =1051 TSC06: MOV      A,R7      ;CHECK FOR 1 SECOND TIMEOUT
047C 9676    =1052      JNZ      TSC04
              =1053 ;
              =1054 ; TURN LASER OFF & RETURN TO CALLER
              =1055 ;

```

1515-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
GHA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 15

```

5      LOC OBJ      LINE      SOURCE STATEMENT
      0480 83      =1057      RET
      0481 90      =1058 ;
      0482 8478    =1059 TSC07: MOVX  DR0,A      ;RESET FCA SHIFT REG
      =1060      JMP    TSC06
      =1061 ;
10     0484 80      =1062 TSC12: MOVX  A,DR0      ;SHIFT OUT 2ND CHAR FROM FCA
      0485 27      =1063      CLR    A
      0486 8950    =1064      MOV    R1,BTSCBUF
      0488 A1      =1065      MOV    DR1,A
      0489 8499    =1066      JMP    TSC09      ;PUT 00 TO R4 FOR 1ST TWO CHAR'S
      0488 0400    =1067 TSC11: JMP    RSTTRP
      0480 8950    =1068 TSC08: MOV    R1,BTSCBUF
      048F 80      =1069      MOVX  A,DR0      ;START OF DATA CAPTURE BUFFER
      0490 F288    =1070      JBT    TSC11      ;READ 1ST CHAR FROM FCA
      0492 9284    =1071      JBT    TSC12      ;JMP TO POWER UP SEQ IF COMM. FROM MOST
      0494 47      =1072      SWAP   A          ;JMP IF 4 CHAR SEG...1ST 2 CHARS=00
      0495 A1      =1073      MOV    DR1,A
      0496 80      =1074      MOVX  A,DR0
      0497 31      =1075      XCHD   A,DR1      ;READ 2ND CHAR FROM FCA
      0498 F1      =1076      MOV    A,DR1
      0499 8930    =1077 TSC09: MOV    R1,BTSEG1
      0498 01      =1078      XRL    A,DR1
      049C C6A6    =1079      JZ     TSC10      ;COMPARE 1ST 2 CHAR'S WITH TSEG1
      049E 8934    =1080      MOV    R1,BTSEG2   ;JMP IF COMPARED
      04A0 F1      =1081      MOV    A,DR1
      04A1 8950    =1082      MOV    R1,BTSCBUF
      04A3 01      =1083      XRL    A,DR1
      04A4 9681    =1084      JNZ    TSC07      ;COMPARE 1ST 2 CHAR'S WITH TSEG2
      =1085 ;      ;JMP TO RESET FCA S.R. IF NO COMPARE
25     04A6 8951    =1086 TSC10: MOV    R1,BTSCBUF+1
      04A8 8416    =1087      CALL   NEXT4      ;SET UP TO READ NEXT 4 CHAR'S
      =1088 ;      ;READ 3RD-6TH CHAR'S
      04AA 8831    =1089      MOV    R0,BTSEG1+1
      04AC 842A    =1090      CALL   SCHTCH
      =1091 ;      ;COMPARE COMPLETE SEG TO TSEG1 & 2
      =1092      ;COMPARE LAST 4 CHAR'S OF CAPTURED SEG
      04AE C688    =1092      JZ     SEGONE      ;TO TSEG1
      0480 85      =1093      CPL     F1        ;JMP IF SEG 1 COMPARED
      0481 8835    =1094      MOV    R0,BTSEG2+1
      0483 842A    =1095      CALL   SCHTCH
      0485 9678    =1096      JNZ    TSC06      ;JMP IF NO MATCH
      0487 8838    =1097      MOV    R0,BTCHT2+1
      0489 7680    =1098      JF1     INCRN
      0488 8839    =1099 SEGONE: MOV    R0,BTCHT1+1
      0480 17      =1100 INCRN: INC    A          ;JMP IF TSEG 2 MATCHED TO INCREMENT
      048E 60      =1101      ADD    A,DR0      ;SET UP R0 TO INCREMENT TSEG 1 CNTR
      048F A0      =1102      MOV    DR0,A      ;ACC NOW = 01
      04C0 CB      =1103      DEC    R0          ;INC LOW BYTE
      04C1 27      =1104      CLR    A
      04C2 70      =1105      ADDC   A,DR0      ;INC HIGH BYTE IF CARRY
      04C3 A0      =1106      MOV    DR0,A
      04C4 8478    =1107      JMP    TSC06      ;JMP TO CNK FOR 1 SECOND TIMEOUT
      1108 ;*****
40     04C6 A3      1109 TROPCK? MOVP  A,DA
      04C7 83      1110      RET
      0500          1111      ORG     500H
      1112 $      INCLUDE(:F1:NORSE3.SRC)
      =1113 ;*****
      =1114 ; FILE: NORSE3.SRC NORSE TEST FILE 3 OF 3.
      =1115 ; 07-03-86 09:30 BOB ACT15
      =1116 ;*****
45     =1117 ; ROUTINE: TNSMD
      =1118 ;
      =1119 ; FUNCTION: SEND A BYTE TO MOST. WAITS FOR TRANSMITTER READY.
      =1120 ;
      =1121 ; ENTRY:
      =1122 ; (R1) = BYTE TO SEND (NO PARITY)
      =1123 ;
      =1124 ; EXIT:
50     =1125 ; *(A)
      =1126 ; *(R1)
      =1127 ; *(R4)
      =1128 ;
      0500 0A      =1129 TNSMD: IN     A,P2
      0501 9200    =1130      JBT    TNSMD
      0503 F9      =1131      MOV    A,R1      ;JMP = MOST COMM. REG. NOT READY
      0504 840E    =1132      CALL   CPARTY     ;(A) = BYTE

```

1515-11 MCS-48/LP1-41 MACRO ASSEMBLER, V4.2
 CHAD95 ASSEMBLED 2/22/88 BY BLAISE ISAACS

PAGE 16

5

10

15

20

25

30

35

40

45

50

55

```

LOC OBJ      LINE      SOURCE STATEMENT
0507 53C0     =1134      ANL      A,#000H
0509 49       =1135      ORL      A,R1      ;(A) = FINAL BYTE WITH PARITY
050A B902     =1136      MOV      R1,#0001A
050C 91       =1137      MOVX    R1,A
0500-83      =1138      RET
              =1139      ;*****
              =1140      ; ROUTINE: CPARTY
              =1141      ;
              =1142      ; FUNCTION: COMPUTE PARITY OF ALTERNATE BITS OF BYTE
              =1143      ;
              =1144      ; ENTRY:
              =1145      ;      (A) = BYTE
              =1146      ;
              =1147      ; EXIT:
              =1148      ;      (A) = PARITY
              =1149      ;      B7 = B5 = B3 = B1 = PARITY OF ODD BITS
              =1150      ;      B6 = B4 = B2 = B0 = PARITY OF EVEN BITS
              =1151      ;      *(R4)
              =1152      ;
050E AC       =1153      CPARTY: MOV     R4,A
050F 47       =1154      SWAP    A
0510 0C       =1155      XRL     A,R4
0511 AC       =1156      MOV     R4,A
0512 E7       =1157      RL      A
0513 E7       =1158      RL      A
0514 0C       =1159      XRL     A,R4
0515 83       =1160      RET
              =1161      ;*****
              =1162      ; NEXT4: READ 3RD-6TH CHARACTERS
              =1163      ;
0516 80       =1164      NEXT4: MOV     A,R0      ;READ 3RD CHAR FROM LS13
0517 47       =1165      SWAP    A
0518 A1       =1166      MOV     R1,A      ;PUT 3RD TO HI NIB. OF TSCBUF+1
0519 80       =1167      MOVX    A,R0      ;READ 4TH CHAR
051A 31       =1168      XCHD    A,R1      ;PUT 4TH TO LO TSCBUF+1
051B 19       =1169      INC     R1      ;POINTS TO TSCBUF+2 NOW
051C 80       =1170      MOVX    A,R0      ;READ 5TH CHAR
051D 47       =1171      SWAP    A
051E A1       =1172      MOV     R1,A      ;PUT 5TH TO HI TSCBUF+2
051F 80       =1173      MOVX    A,R0      ;READ 6TH CHAR
0520 31       =1174      XCHD    A,R1      ;PUT 6TH TO LO TSCBUF+2
0521 19       =1175      INC     R1
0522 B804     =1176      MOV     R0,#04H
0524 80       =1177      MOVX    A,R0      ;R0=04H,R0=PARITY DECODE READ OF LS13
0525 A1       =1178      MOV     R1,A      ;PUT PARITY DECODE TO TSCBUF+3
0526 B801     =1179      MOV     R0,#01H
0528 90       =1180      MOVX    R0,A      ;FR.RST TO LSI 3
0529 83       =1181      RET
              =1182      ;*****
              =1183      ; ROUTINE: SGMTCN - MATCH SEGMENTS
              =1184      ;
              =1185      ; FUNCTION: DETERMINE IF TWO SEGMENTS HAVE IDENTICAL DATA AND PARITY
              =1186      ;      (LAST 4 CHARACTERS OF SEG AND PARITY)
              =1187      ;
              =1188      ; ENTRY:
              =1189      ;      (R0) = ADDR OF ONE SEGMENT - SAME FORMAT AS REG 4-7
              =1190      ;      (TSCBUF+1) = 3RD AND 4TH CHAR
              =1191      ;      (TSCBUF+2) = 5TH AND 6TH CHAR
              =1192      ;      (TSCBUF+3) = DECODED PARITY WORD
              =1193      ;
              =1194      ; EXIT:
              =1195      ;      (A) = 0 FOR MATCH
              =1196      ;      *(R0)
              =1197      ;
052A F0       =1198      SGMTCN: MOV     A,R0
052B B951     =1199      MOV     R1,#TSCBUF+1
052D 01       =1200      XRL     A,R1
052E 963A     =1201      JNZ     SGN90      ;JMP = 2ND BYTE NO MATCH
0530 18       =1202      INC     R0
0531 19       =1203      INC     R1
0532 F0       =1204      MOV     A,R0
0533 01       =1205      XRL     A,R1
0534 963A     =1206      JNZ     SGN90      ;JMP = 3RD BYTE NO MATCH
0536 18       =1207      INC     R0
0537 19       =1208      INC     R1
0538 F0       =1209      MOV     A,R0

```

5 1S1S-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2 PAGE 17
QNA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

LOC	OBJ	LINE	SOURCE STATEMENT
	053A 83	=1211	SCMP0: RET
		=1212	*****
10		=1213	ROUTINE: THWAIT
		=1214	*****
		=1215	FUNCTION: WAIT FOR 20MSEC TIMER TO EXHAUST
		=1216	*****
		=1217	ENTRY:
		=1218	RBO (R7) = TIMER (20MSEC RES)
		=1219	*****
		=1220	EXIT:
		=1221	RBO (R7) = 0
		=1222	*****
15	0538 27	=1223	THWAIT: CLR A
	053C 62	=1224	MOV T,A
	053D 25	=1225	EN TCHTI ;CLEAR THE TIMER COUNTER
	053E 55	=1226	STRT T ;ENABLE TIMER INTERRUPT
		=1227	***** ;CLEAR TIMER PRESCALER
	053F FF	=1228	THWAIT: MOV A,R7
	0540 963F	=1229	JNZ THWAIT
20	0542 83	=1230	RET
		1231	*****
	0543 A3	1232	TROPCS: MOVP A,3A
	0544 83	1233	RET
	0600	1234	ORG 600H
		1235	\$ INCLUDE(:F1:FCKCTS.SRC)
		=1236	*****
25		=1237	FILE: FCKCTS.SRC 10-08-86 15:45 BOB ACTIS
		=1238	ROUTINE: CCKCTS
		=1239	FUNCTION: CHECK SEGMENT BUFFER TOTAL COUNTS FOR ENOUGH SEGMENTS FOR
		=1240	A POSSIBLE VALID VERSION.
		=1241	ENTRY: NO SETUP
		=1242	EXIT: A = 0 IF ENOUGH SEGMENTS
		=1243	A < 0 IF NOT ENOUGH SEGMENTS OR (VERSION D IF IBM-OCR 1/F)
		=1244	USES R0,R1,F0
		=1245	*****
30	0600 B9FE	=1246	CCKCTS: MOV R1,#-2 ;SETUP MINIMUM SCANS REQUIRED VALUE
		=1247	*****
	0602 85	=1248	CLR F0
	0603 0A	=1249	IN A,P2
	0604 37	=1250	CPL A
	0605 F208	=1251	JBT CCKC05
	0607 95	=1252	CPL F0 ;JUMP IF FUJITSU 1/F
		=1253	***** ;SET F0 IF IBM-OCR 1/F
35	0608 B835	=1254	CCKC05: MOV R0,#R6STOT
	060A F0	=1255	MOV A,R0
	0608 69	=1256	ADD A,R1
	060C F622	=1257	JC CCKC20
		=1258	***** ;POSSIBLE UPC-A, EAN-13 OR UPC-D-BLK2
	060E B82D	=1259	MOV R0,#L6STOT
	0610 F0	=1260	MOV A,R0
40	0611 69	=1261	ADD A,R1
	0612 F62F	=1262	JC CCKC30
		=1263	***** ;POSSIBLE UPC-E OR UPC-D-BLK1
	0614 B838	=1264	MOV R0,#L4STOT
	0616 F0	=1265	MOV A,R0
	0617 69	=1266	ADD A,R1
	0618 E695	=1267	JNC CCKCNG
		=1268	*****
45	061A B841	=1269	CCKC10: MOV R0,#R4STOT
	061C F0	=1270	MOV A,R0
	061D 69	=1271	ADD A,R1
	061E F693	=1272	JC CCKCOK
	0620 C495	=1273	JMP CCKCNG ;POSSIBLE EAN-8
		=1274	*****
	0622 B82D	=1275	CCKC20: MOV R0,#L6STOT
	0624 F0	=1276	MOV A,R0
50	0625 69	=1277	ADD A,R1
	0626 E695	=1278	JNC CCKCNG
		=1279	*****
	0628 B827	=1280	MOV R0,#L6S1+3
	062A F0	=1281	MOV A,R0
	062B B24F	=1282	JBS CCKC50
	062D C493	=1283	JMP CCKCOK ;GET THE PARITY DECODE BYTE
		=1284	***** ;JUMP IF UPC-D SEGMENT
55	062F B827	=1285	CCKC30: MOV R0,#L6S1+3
	0631 F0	=1286	MOV A,R0 ;CHECK L6 BUFFER 1

IS15-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2 PAGE 18
QNA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

LOC	OBJ	LINE	SOURCE STATEMENT	
	0634 9293	=1288	JB4 CXCNOK	;POSSIBLE UPC-E
		=1289 ;		
	0636 8828	=1290	MOV R0,#L6S2+3	;CHECK L6 BUFFER 2
	0638 F0	=1291	MOV A,DRO	
10	0639 823F	=1292	JB5 CXCNO40	;JUMP IF UPC-D SEGMENT
	0638 9293	=1293	JB4 CXCNOK	;POSSIBLE UPC-E
	0630 C495	=1294	JMP CXCNOG	
		=1295 ;		
	063F 8865	=1296	CXCNO40: MOV R0,#N6STOT	
	0641 F0	=1297	MOV A,DRO	
	0642 69	=1298	ADD A,R1	
	0643 E695	=1299	JNC CXCNOG	
15		=1300 ;		
	0645 8838	=1301	CXCNO45: MOV R0,#L4STOT	
	0647 F0	=1302	MOV A,DRO	
	0648 69	=1303	ADD A,R1	
	0649 8695	=1304	JFO CXCNOG	;JUMP IF IBM-OCR
	0648 F693	=1305	JC CXCNOK	;POSSIBLE UPC-D1
	0640 C495	=1306	JMP CXCNOG	
		=1307 ;		
20	064F 8841	=1308	CXCNO50: MOV R0,#N4STOT	
	0651 F0	=1309	MOV A,DRO	
	0652 69	=1310	ADD A,R1	
	0653 E695	=1311	JNC CXCNOG	
		=1312 ;		
	0655 8859	=1313	MOV R0,#N4STOT	
	0657 F0	=1314	MOV A,DRO	
25	0658 69	=1315	ADD A,R1	
	0659 F675	=1316	JC CXCNO70	;POSSIBLE UPC-D4 OR D5
		=1317 ;		
	0658 885F	=1318	MOV R0,#N5STOT	
	0650 F0	=1319	MOV A,DRO	
	065E 69	=1320	ADD A,R1	
	065F F668	=1321	JC CXCNO60	
		=1322 ;		
30	0661 8840	=1323	MOV R0,#N2STOT	
	0663 F0	=1324	MOV A,DRO	
	0664 69	=1325	ADD A,R1	
	0665 8695	=1326	JFO CXCNOG	;JUMP IF IBM-OCR
	0667 F693	=1327	JC CXCNOK	;POSSIBLE UPC-D2
	0669 C495	=1328	JMP CXCNOG	
		=1329 ;		
	0668 8853	=1330	CXCNO60: MOV R0,#N3STOT	
35	0660 F0	=1331	MOV A,DRO	
	066E 69	=1332	ADD A,R1	
	066F 8695	=1333	JFO CXCNOG	;JUMP IF IBM-OCR
	0671 F693	=1334	JC CXCNOK	;POSSIBLE UPC-D3
	0673 C495	=1335	JMP CXCNOG	
		=1336 ;		
	0675 8847	=1337	CXCNO70: MOV R0,#N1STOT	
40	0677 F0	=1338	MOV A,DRO	
	0678 69	=1339	ADD A,R1	
	0679 E695	=1340	JNC CXCNOG	
		=1341 ;		
	0678 8865	=1342	MOV R0,#N6STOT	
	0670 F0	=1343	MOV A,DRO	
	067E 69	=1344	ADD A,R1	
	067F F688	=1345	JC CXCNO80	
		=1346 ;		
45	0681 885F	=1347	MOV R0,#N5STOT	
	0683 F0	=1348	MOV A,DRO	
	0684 69	=1349	ADD A,R1	
	0685 8695	=1350	JFO CXCNOG	;JUMP IF IBM-OCR
	0687 F693	=1351	JC CXCNOK	;POSSIBLE UPC-D4
	0689 C495	=1352	JMP CXCNOG	
		=1353 ;		
50	0688 8853	=1354	CXCNO80: MOV R0,#N3STOT	
	0680 F0	=1355	MOV A,DRO	
	068E 69	=1356	ADD A,R1	
	068F 8695	=1357	JFO CXCNOG	;JUMP IF IBM-OCR
	0691 E695	=1358	JNC CXCNOG	;FALL THRU POSSIBLE UPC-D5
		=1359 ;		
	0693 27	=1360	CXCNOK: CLR A	;ENOUGH SCANS FOR A POSSIBLE SEGMENT
	0694 83	=1361	RET	
		=1362 ;		
55	0695 27	=1363	CXCNOG: CLR A	;NO POSSIBLE VERSIONS YET

ISIS-11 MCS-48/UPI-41 MACRO ASSEMBLER, V4.2
 QWAD95 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 19

```

5      LOC OBJ      LINE      SOURCE STATEMENT
      0697 83      =1365      RET
                        1366 $      INCLUDE(:F1:COPYRT.SRC)
                        =1367 ;
                        =1368 ;
10      0698 434F5059 =1369      DB      'COPYRIGHT (C)1987 '
      069C 52494748
      06A0 54202843
      06A4 29313938
      06A8 3720
      06AA 53504543 =1370      DB      'SPECTRA-PHYSICS, INC. '
      06AE 54524120
      06B2 50485953
      06B6 4943532C
      06BA 20494E43
15      06BE 2E20
      06C0 414C4C20 =1371      DB      'ALL RIGHTS RESERVED'
      06C4 52494748
      06C8 54532052
      06CC 45534552
      06D0 564544

20      =1372 ;
      =1373 ;
      1374 ;
      06D3 A3      1375 TROP66: MOVP  A,BA
      06D4 83      1376      RET
      0700      1377      ORG      700H
                        1378 $      INCLUDE(:F1:DRSUNT.SRC)
                        =1379 ;
25      =1380 ; FILE: DRSUNT 05-28-86 11:00 BOB ACTIS
      =1381 ;
      =1382 ; ROUTINE: SUM4BY, SUM3BY, SUM2BY
      =1383 ; FUNCTION: ADD 4,3 OR 2 BYTE SEGMENT DATA TO ACCUMULATOR
      =1384 ; ENTRY: R0 = START ADDRESS OF SEGMENT BUFFER TO SUM
      =1385 ; EXIT: R0 = END ADDRESS OF SEGMENT BUFFER
      =1386 ; A = RUNNING SUM (BASE 256) OF SEGMENT BUFFER DATA
      =1387 ;
30      0700 60      =1388 SUM4BY: ADD  A,R0
      0701 18      =1389      INC  R0
      0702 60      =1390 SUM3BY: ADD  A,R0
      0703 18      =1391      INC  R0
      0704 60      =1392 SUM2BY: ADD  A,R0
      0705 18      =1393      INC  R0
      0706 60      =1394      ADD  A,R0
      0707 83      =1395      RET
35      =1396 ;
      =1397 ; ROUTINE: DRSUNT DOUBLE READ SUM TEST
      =1398 ; FUNCTION: CALCULATE THE LABEL DATA SUM (BASE 256)
      =1399 ; COMPARE IT TO THE PREVIOUS LABEL SUM
      =1400 ; SAVE THE NEW SUM
      =1401 ; ENTRY: A VALID LABEL VERSION HAS BEEN FOUND
      =1402 ; EXIT: USES R0
40      =1403 ; LABEL DATA SUM STORED IN DRSUM
      =1404 ; A=0 IF OLD=NEW (CONSECUTIVE LABELS ARE THE SAME)
      =1405 ; A<0 IF OLD<NEW (CONSECUTIVE LABELS ARE DIFFERENT)
      =1406 ;
      =1407 DRSUNT: MOV  A,R6      ;GET VERSION FLAG
      0708 FE      =1408      ANL  A,#0FH      ;MASK VERSION POINTER
      0709 530F    =1409      ADD  A,R0      ;SETUP CARRY FOR BA
      070A 0300
      070D 57      =1410      BA  A
      070E 926C    =1411      J84  DRYING      ;JUMP IF POINTER > 9. ILLEGAL VERSION.
45      =1412 ;
      0710 8313    =1413      ADD  A,#LOW DRSSTBL ;ADD OFFSET TO LABEL ADDRESS
      0712 83      =1414      JMP  BA      ;JUMP TO VERSION ROUTINE
      0713 6C      =1415 DRSSTBL: DB  LOW DRYING ;NO VALID VERSION. POINTER=0.
      0714 10      =1416      DB  LOW DRSNA
      0715 10      =1417      DB  LOW DRSN13
      0716 20      =1418      DB  LOW DRSNE
      0717 5E      =1419      DB  LOW DRSN3
50      0718 42      =1420      DB  LOW DRSND1
      0719 48      =1421      DB  LOW DRSND2
      071A 39      =1422      DB  LOW DRSND3
      071B 2C      =1423      DB  LOW DRSND4
      071C 23      =1424      DB  LOW DRSND5
      =1425 ;
      071D      =1426 DRSNA EQU  S
55      071D 27    =1427 DRSN13: CLR  A

```

5

 ISIS-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
 QMA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 20

10

15

20

25

30

35

40

45

50

55

LOC	OBJ	LINE	SOURCE STATEMENT
		=1429 ;	
0720	27	=1430 DRSH7C: CLR	A
0721	E458	=1431 JMP	DRSH7C ;GO SUM 7 CHARACTERS
		=1432 ;	
0723	27	=1433 DRSH05: CLR	A
0724	B860	=1434 MOV	R0,#N6S1
0726	F404	=1435 CALL	SUM2BY
0728	B84E	=1436 MOV	R0,#N3S1
072A	E42F	=1437 JMP	DRSH5X ;GO FINISH THE 05 SUM
		=1438 ;	
072C	27	=1439 DRSH04: CLR	A
072D	B85A	=1440 MOV	R0,#N5S1
072F	F404	=1441 DRSH5X: CALL	SUM2BY
0731	B854	=1442 MOV	R0,#N4S1
0733	F404	=1443 CALL	SUM2BY
0735	B842	=1444 MOV	R0,#N1S1
0737	E44E	=1445 JMP	DRSH4X ;GO FINISH THE 04 OR 05 SUM
		=1446 ;	
0739	27	=1447 DRSH03: CLR	A
073A	B85A	=1448 MOV	R0,#N5S1
073C	F404	=1449 CALL	SUM2BY
073E	B84E	=1450 MOV	R0,#N3S1
0740	E44E	=1451 JMP	DRSH3X ;GO FINISH THE 03 SUM
		=1452 ;	
0742	27	=1453 DRSH01: CLR	A
0743	B860	=1454 MOV	R0,#N6S1
0745	F404	=1455 CALL	SUM2BY
0747	B836	=1456 MOV	R0,#L4S1
0749	E456	=1457 JMP	DRSH1X ;GO FINISH THE 01 SUM
		=1458 ;	
074B	27	=1459 DRSH02: CLR	A
074C	B848	=1460 MOV	R0,#N2S1
074E		=1461 DRSH5X: EQU	5
074E	F404	=1462 DRSH4X: CALL	SUM2BY
0750	B83C	=1463 MOV	R0,#N4S1
0752	F404	=1464 CALL	SUM2BY
0754	B82E	=1465 DRSH7B: MOV	R0,#N6S1 ;SUM 7 BYTES
0756	F402	=1466 DRSH1X: CALL	SUM3BY
0758	B824	=1467 DRSH7C: MOV	R0,#L6S1 ;SUM 7 CHARACTERS
075A	F400	=1468 CALL	SUM4BY
075C	E467	=1469 JMP	DRSH0N ;JUMP - THE SUM IS FINISHED
		=1470 ;	
075E	27	=1471 DRSH8: CLR	A
075F	B836	=1472 MOV	R0,#L4S1
0761	F404	=1473 CALL	SUM2BY
0763	B83C	=1474 MOV	R0,#N4S1
0765	F404	=1475 CALL	SUM2BY
		=1476 ;	
0767	B87C	=1477 DRSH0N: MOV	R0,#DRSUM
0769	20	=1478 XCH	A,DR0
076A	00	=1479 XRL	A,DR0 ;SAVE THE NEW SUM IN DRSUM
076B	83	=1480 RET	;COMPARE THE OLD AND NEW SUM
		=1481 ;	
076C	27	=1482 DRVRNG: CLR	A ;DOUBLE READ VERSION N/C
076D	83	=1483 RET	
		=1484 ;	
076E	A3	=1485 TROPCT: MOV	A,DA
076F	83	=1486 RET	
0800		=1487 ORG	800H ;START OF MEMORY BANK 1
		=1488 \$	INCLUDE(=F1:VERDLB.SRC)
		=1489 ;	
		=1490 ;	FILE: VERDLB.SRC 6-17-86 17:15 BOB ACTIS
		=1491 ;	VERSION "D" FIRMWARE LIBRARY.
		=1492 ;	
		=1493 ;	ROUTINE: CLR6SG
		=1494 ;	FUNCTION: CLEAR 6-CHAR SEGMENTS AND COUNTERS.
		=1495 ;	ENTRY: START ADDRESS AND COUNT IN DEFS TABLE.
		=1496 ;	EXIT: A="0"
		=1497 ;	R0 = END OF 6-CHAR BUFFER/COUNTER SPACE PLUS 1.
		=1498 ;	R2 = 0
		=1499 ;	6-CHAR BUFFER/COUNTER SPACE = 0'S.
		=1500 ;	
0800	B824	=1501 CLR6SG: MOV	R0,#BF6CST ;START OF 6-CHAR BUFFER AREA
0802	8A12	=1502 MOV	R2,#BF6CNT ;NUMBER OF BYTES IN BUFFER
0804	0410	=1503 JMP	CLRT00 ;JUMP TO THE CLEAR LOOP
		=1504 ;	

1515-II MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
QMA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 21

LOC	OBJ	LINE	SOURCE STATEMENT
		=1506 ;	FUNCTION: CLEAR 4-CHAR SEGMENTS AND COUNTERS.
		=1507 ;	ENTRY: START ADDRESS AND COUNT IN DEFS TABLE.
		=1508 ;	EXIT: A = 0
10		=1509 ;	R0 = END OF 4-CHAR BUFFER/COUNTER SPACE PLUS 1.
		=1510 ;	R2 = 0
		=1511 ;	4-CHAR BUFFER/COUNTER SPACE = 0'S.
		=1512 ;	
0806 8836		=1513 CLR4SG:	MOV R0,#BF4CST ;START OF 4-CHAR BUFFER AREA
0808 BA30		=1514	MOV R2,#BF4CNT ;NUMBER OF BYTES IN BUFFER
080A 0410		=1515	JMP CLRT00 ;JUMP TO THE CLEAR LOOP
		=1516 ;	ROUTINE: CLRSNB
15		=1517 ;	FUNCTION: CLEAR THE SCAN BUFFER.
		=1518 ;	ENTRY: START ADDRESS IN DEFS TABLE.
		=1519 ;	EXIT: A = 0
		=1520 ;	R0 = END OF SCAN BUFFER SPACE PLUS 1.
		=1521 ;	R2 = 0
		=1522 ;	SCAN BUFFER SPACE = 0'S.
		=1523 ;	
		=1524 ;	
20	080C 8820	=1525 CLRSNB:	MOV R0,#SCANBF ;START OF SCAN BUFFER AREA
	080E BA04	=1526	MOV R2,#4 ;NUMBER OF BYTES IN BUFFER
	0810 27	=1527 CLRT00:	CLR A ;CLEAR LOOP USED BY OTHER ROUTINES
	0811 A0	=1528 CLRSN1:	MOV R0,A
	0812 18	=1529	INC R0
	0813 EA11	=1530	DJNZ R2,CLRSN1
	0815 83	=1531	RET
		=1532 ;	ROUTINE: CLRSBF
25		=1533 ;	FUNCTION: CLEAR SEND BUFFER, POINTER AND FLAG.
		=1534 ;	ENTRY: START ADDRESS AND COUNT IN DEFS TABLE.
		=1535 ;	R80
		=1536 ;	EXIT: A = 0
		=1537 ;	R0 = END OF SEND BUFFER SPACE PLUS 1.
		=1538 ;	R2 = 0
30		=1539 ;	SEND BUFFER SPACE = 0CCH'S. (TERMINATION BYTES)
		=1540 ;	SEND BUFFER FULL FLAG CLEAR. R80-R4-R3
		=1541 ;	SEND BUFFER POINTER SET TO PACKED DATA START ADDRESS.
		=1542 ;	
		=1543 ;	
	0816 FC	=1544 CLRSBF:	MOV A,R4
	0817 53F7	=1545	ANL A,#255-ESBFUL ;CLEAR SEND BUFFER FULL FLAG
	0819 AC	=1546	MOV R4,A
		=1547 ;	
35	081A 8866	=1548	MOV R0,#SBPNT
	081C 80CE	=1549	MOV R0,#SBSTRT ;SET POINTER TO PACKED START ADDRESS
		=1550 ;	
	081E 8867	=1551	MOV R0,#SBUF
	0820 BA12	=1552	MOV R2,#SBUSZ ;START OF SEND BUFFER AREA
	0822 23CC	=1553	MOV A,#0CCH ;NUMBER OF BYTES IN BUFFER
	0824 0411	=1554	JMP CLRSN1 ;LOAD TERMINATION BYTES
		=1555 ;	JUMP TO THE CLEAR LOOP
40		=1556 ;	ROUTINE: MOV2BT, MOV3BT, MOV4BT
		=1557 ;	FUNCTION: MOVE BYTES FROM ONE BUFFER TO ANOTHER BUFFER.
		=1558 ;	ENTRY: R0 = FIRST BYTE ADDRESS OF SOURCE BUFFER.
		=1559 ;	R1 = FIRST AVAILABLE BYTE ADDRESS OF DESTINATION BUFFER.
		=1560 ;	EXIT: DATA MOVED FROM SOURCE BUFFER TO DESTINATION BUFFER.
		=1561 ;	R0 = END OF SOURCE BUFFER ADDRESS PLUS 1.
		=1562 ;	R1 = NEXT AVAILABLE BYTE ADDRESS OF DESTINATION BUFFER.
		=1563 ;	R2 = 0
45		=1564 ;	A = LAST BYTE TRANSFERED
		=1565 ;	
	0826 BA02	=1566 MOV2BT:	MOV R2,R2
	0828 0430	=1567	JMP MOVXBT
	082A BA03	=1568 MOV3BT:	MOV R2,R3
	082C 0430	=1569	JMP MOVXBT
	082E BA04	=1570 MOV4BT:	MOV R2,R4
	0830 F0	=1571 MOVXBT:	MOV A,R0
50	0831 A1	=1572	MOV R0,A
	0832 18	=1573	INC R0
	0833 19	=1574	INC R1
	0834 EA30	=1575	DJNZ R2,MOVXBT
	0836 83	=1576	RET
		=1577 ;	ROUTINE: SCSUM4, SCSUM6
55		=1578 ;	FUNCTION: SUM THE DIGITS OF A SEGMENT FOR THE MOD-10 TEST.
		=1579 ;	ALL ARITHMETIC IS ASSUMED BCD AND ONLY THE UNITS DIGIT
		=1580 ;	IS VALID IN THE SUMS.
		=1581 ;	

1815-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
 QMAD95 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 22

5

10

15

20

25

30

35

40

45

50

55

```

LOC OBJ      LINE      SOURCE STATEMENT
=1583 ; EXIT: R0 = NOT CHANGED
=1584 ; R2 = C1+C3+C5 [C1+C3]
=1585 ; A = C2+C4+C6 [C2+C4]
=1586 ; F0 IS USED
=1587 ;
0837 85      =1588 SCSUM4: CLR    F0
0838 95      =1589          CPL    F0          ;SET F0 FOR 4-CHAR SUM
0839 043C    =1590          JMP    SCSUMX
=1591 ;
0838 85      =1592 SCSUM6: CLR    F0          ;CLEAR F0 FOR 6-CHAR SUM
083C BAFO    =1593 SCSUMX: MOV    R2,#0F0H    ;MASK FOR ODD DIGITS
083E F0      =1594          MOV    A,R0      ;C1
083F 5A      =1595          ANL    A,R2
0840 18      =1596          INC    R0
0841 60      =1597          ADD    A,R0      ;C1+C3
0842 57      =1598          DA     A
0843 5A      =1599          ANL    A,R2
0844 B649    =1600          JF0    SCSUMY    ;JUMP IF 4-CHAR SUM
=1601 ;
0846 18      =1602          INC    R0
0847 60      =1603          ADD    A,R0      ;C1+C3+C5
0848 57      =1604          DA     A
0849 47      =1605 SCSUMY: SWAP   A          ;PUT SUM IN LOW BYTE
084A AA      =1606          MOV    R2,A      ;SAVE C1+C3+C5 [C1+C3]
=1607 ;
084B F0      =1608          MOV    A,R0      ;C6 [C4]
084C C8      =1609          DEC    R0
084D 60      =1610          ADD    A,R0      ;C6+C4 [C4+C2]
084E 57      =1611          DA     A
084F B654    =1612          JF0    SCSUMR    ;JUMP IF 4-CHAR SUM
=1613 ;
0851 C8      =1614          DEC    R0
0852 60      =1615          ADD    A,R0      ;C6+C4+C2
0853 57      =1616          DA     A
0854 83      =1617 SCSUMR: RET
=1618 ;
=1619 ; ROUTINE: APL3R2
=1620 ; FUNCTION: ADD 3*R2 TO A. (BCD)
=1621 ; ENTRY: NO SETUP
=1622 ; EXIT: A = A+(3*R2)
=1623 ; R2 = NOT CHANGED
=1624 ;
0855 6A      =1625 APL3R2: ADD    A,R2
0856 57      =1626          DA     A
0857 6A      =1627          ADD    A,R2
0858 57      =1628          DA     A
0859 6A      =1629          ADD    A,R2
085A 57      =1630          DA     A
085B 83      =1631          RET
=1632 ;
=1633 ; ROUTINE: MOD104 6-2-86 14:25 BOB ACTIS
=1634 ; FUNCTION: CALCULATE 4 CHARACTER MODULO 10 CHECKSUM VALUE
=1635 ; ENTRY: R0 = FIRST BYTE ADDRESS OF SEGMENT TO BE PROCESSED
=1636 ; EXIT: A = CALCULATED VALUE
=1637 ;
085C 1437    =1638 MOD104: CALL    SCSUM4
085E 1455    =1639          CALL    APL3R2
0860 83      =1640          RET
=1641 ;
=1642 ; ROUTINE: MOD106 6-2-86 14:25 BOB ACTIS
=1643 ; FUNCTION: CALCULATE 6 CHARACTER MODULO 10 CHECKSUM VALUE
=1644 ; ENTRY: R0 = FIRST BYTE ADDRESS OF SEGMENT TO BE PROCESSED
=1645 ; EXIT: A = CALCULATED VALUE
=1646 ;
0861 1438    =1647 MOD106: CALL    SCSUM6
0863 1455    =1648          CALL    APL3R2
0865 83      =1649          RET
=1650 ; INCLUDE(:"F1:EMOD10.SRC)
=1651 ;
=1652 ; FILE: EMOD10.SRC 6-17-86 16:25 BOB ACTIS
=1653 ; ROUTINE: EMOD10
=1654 ; FUNCTION: VERIFY THE EMOD10 CHECK CHARACTER
=1655 ; ENTRY: SEGMENT IN SCAN BUFFER
=1656 ; EXIT: USES R0,R1,R2,A
=1657 ; A = 0 IF CHECK CHARACTER IS GOOD
=1658 ; A <> 0 IF CHECK CHARACTER IS BAD

```

5 ISIS-II MCS-48/UP1-41 MACRO ASSEMBLER, V4.2 PAGE 23
 QHA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

LOC	OBJ	LINE	SOURCE STATEMENT
	0866 8820	=1660	EMOD10: MOV R0,#SCANBUF
	0868 8979	=1661	MOV R1,#WKBUFF
	086A 142A	=1662	CALL MOV3BY
10		=1663	; MOVE SCAN BUFFER DATA TO WORK BUFFER
	086C 8822	=1664	MOV R0,#SCANBUF+2
	086E F0	=1665	MOV A,BRO
	086F 530F	=1666	ANL A,BROFN
	0871 03FD	=1667	ADD A,#-3
	0873 F27C	=1668	JBT ECASE1
	0875 C685	=1669	JZ ECASE2
	0877 07	=1670	DEC A
15	0878 C690	=1671	JZ ECASE3
	087A 0496	=1672	JMP ECASE4
		=1673	; JUMP IF C6=4
		=1674	; JUMP IF C6=5,6,7,8,9
		=1675	; FOR C6=0,1,2 COMPUTE 3*(C2+C3+C5)+C1+C4+C6
	087C 8879	=1676	ECASE1: MOV R0,#WKBUFF
	087E F0	=1677	MOV A,BRO
	087F 47	=1678	SWAP A
20	0880 A0	=1679	MOV BRO,A
	0881 1438	=1680	CALL SGSUM6
	0883 0498	=1681	JMP EMODSH
		=1682	; C2,C1,C3,C4,C5,C6
		=1683	; FOR C6=3 COMPUTE 3*(C2+0+C5)+C1+C3+C4
		=1684	; ECASE2: MOV R0,#WKBUFF+1
25	0885 887A	=1685	CLR A
	0887 27	=1686	XCHD A,BRO
	0888 30	=1687	INC R0
	0889 18	=1688	XCHD A,BRO
	088A 30	=1689	MOV A,BRO
	088B F0	=1690	SWAP A
	088C 47	=1691	MOV BRO,A
	088D A0	=1692	JMP ECASE4
	088E 0496	=1693	; C1,C2,C3,0,C4,C5
30		=1694	; FOR C6=4 COMPUTE 3*(C2+C4+C5)+C1+C3+0
		=1695	; ECASE3: MOV R0,#WKBUFF+2
	0890 887B	=1696	CLR A
	0892 27	=1697	XCHD A,BRO
	0893 20	=1698	SWAP A
	0894 47	=1699	MOV BRO,A
	0895 30	=1700	XCHD A,BRO
		=1701	; C5=0, C6=C5
		=1702	; C1,C2,C3,C4,0,C5
35		=1703	; FOR C6=5,6,7,8,9 COMPUTE 3*(C2+C4+C6)+C1+C3+C5
		=1704	; ECASE4: MOV R0,#WKBUFF
	0896 8879	=1705	CALL SGSUM6
	0898 1438	=1706	XCHD A,R2
	089A 2A	=1707	; C1,C2,C3,C4,C5,C6
		=1708	; FINAL SUM
40	089B 1455	=1709	CALL APLSR2
		=1710	; CHECK AGAINST THE CHECK CHARACTER
		=1711	; EMODCK: MOV R0,#SCANBUF+3
	089D 8823	=1712	ADD A,BRO
	089F 60	=1713	DA A
	08A0 57	=1714	ANL A,BROFN
45	08A1 530F	=1715	RET
	08A3 83	=1716	INCLUDE(:F1:CKFCFCA.SRC)
		=1717	*****
		=1718	FILE: CKFCFCA.SRC 10-25-83 17:10 BOB ACTIS
		=1719	ROUTINE: CKFCA
		=1720	FUNCTION: CHECK IF FRAME CONTROL ARRAY HAS DATA.
50		=1721	IF SDATA, PROCESS BYTE.
		=1722	IF SEGMENT CAPTURE AND SCANNING BIT IS SET, PUT SEGMENT
		=1723	INTO THE SCAN BUFFER.
		=1724	IF SEGMENT CAPTURE AND NOT SCANNING, RESET THE FRAME.
		=1725	IF A SEGMENT IS SEEN, SET R7.
		=1726	ENTRY: R0
		=1727	EXIT: USES R0,R1,A
		=1728	R7 IS SET IF A SEGMENT IS SEEN.
55	08A4 86A7	=1729	JNI CKFC10
	08A6 83	=1730	RET
		=1731	; JUMP IF FCA HAS DATA

ISIS-II MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
 QMA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 24

LOC	OBJ	LINE	SOURCE STATEMENT
5	08A7 B801	=1737	CKFC10: MOV R0, #RESRD
	08A9 FC	=1738	MOV A, R4
	08AA 1283	=1739	J80 CKFC20 ;JUMP IF SCANNING BIT IS SET
	08AC 80	=1740	MOVX A, DR0 ;READ BYTE
	08AD F2DF	=1741	J87 CKFC95 ;JUMP IF MOST DATA
	08AF 90	=1742	MOVX DR0, A ;RESET FRAME CAPTURE
10	08B0 8FD8	=1743	MOV R7, #WAIT ;SET THE "SEG SEEN" TIMER
	08B2 83	=1744	RET
		=1745 ;	
	08B3 80	=1746	CKFC20: MOVX A, DR0 ;READ 1ST CHAR OF SEGMENT
	08B4 F2DF	=1747	J87 CKFC95 ;JUMP IF MOST DATA
	08B6 02DB	=1748	J86 CKFC90 ;JUMP IF PERIODICAL CAPTURE
	08B8 8920	=1749	MOV R1, #SCNBUF
		=1750 ;	
15	08BA 47	=1751	SWAP A
	08BB A1	=1752	MOV DR1, A ;STORE 1ST CHAR
	08BC 80	=1753	MOVX A, DR0 ;READ 2ND CHAR OF SEGMENT
	08BD 31	=1754	XCHD A, DR1 ;1ST AND 2ND STORED
	08BE 19	=1755	INC R1
	08BF 80	=1756	MOVX A, DR0 ;READ 3RD CHAR
		=1757 ;	
20	08C0 47	=1758	SWAP A
	08C1 A1	=1759	MOV DR1, A ;STORE 3RD CHAR
	08C2 80	=1760	MOVX A, DR0 ;READ 4TH CHAR
	08C3 31	=1761	XCHD A, DR1 ;3RD AND 4TH STORED
	08C4 19	=1762	INC R1
	08C5 80	=1763	MOVX A, DR0 ;READ 5TH CHAR
		=1764 ;	
25	08C6 47	=1765	SWAP A
	08C7 A1	=1766	MOV DR1, A ;STORE 5TH CHAR
	08C8 80	=1767	MOVX A, DR0
	08C9 31	=1768	XCHD A, DR1 ;5TH AND 6TH STORED
	08CA 19	=1769	INC R1
		=1770 ;	
30	08CB 8804	=1771	MOV R0, #PRDEC
	08CD 80	=1772	MOVX A, DR0 ;READ PARITY DECODE BYTE
	08CE 537F	=1773	ANL A, #07FH ;MASK OFF THE UNUSED BIT
	08D0 A1	=1774	MOV DR1, A ;STORE PARITY DECODE 80-86
		=1775 ;	
	08D1 837F	=1776	XRL A, #07FH ;TEST FOR NO DECODE. (BAD PARITY MAP)
	08D3 9609	=1777	JNZ CKFC80 ;JUMP IF DECODE IS OK.
	08D5 140C	=1778	CALL CLRSHB ;CLEAR THE SCAN BUFFER
	08D7 0408	=1779	JMP CKFC90
		=1780 ;	
35	08D9 8FD8	=1781	CKFC80: MOV R7, #WAIT ;SET THE "SEG SEEN" TIMER
		=1782 ;	
	08DB 8801	=1783	CKFC90: MOV R0, #PERST
	08DD 90	=1784	MOVX DR0, A ;RESET THE FRAME
	08DE 83	=1785	RET
		=1786 ;	
40	08DF E5	=1787	CKFC95: SEL M80
	08E0 143F	=1788	CALL SDATA
	08E2 F5	=1789	SEL M81
	08E3 83	=1790	RET
		=1791 ;	
	08E4 A3	=1792	TROPGB: MOV A, BA
	08E5 83	=1793	RET
	0900	=1794	ORG 0900H
		=1795 ;	INCLUDE(:F1:NDMM.SRC)
		=1796 ;	
45		=1797 ;	FILE: NDMM.SRC 4-16-87 DREV TAUSIG
		=1798 ;	MODIFIED 8-5-87 REMOVE P13 FROM HANDSHAKE
		=1799 ;	ROUTINE: NDMM -- FOR THE IBM 4683 SERIAL IO CHANNEL
		=1800 ;	COMMUNICATES WITH ZILOG SUPER-8 CHANNEL CONTROLLER
		=1801 ;	FUNCTION: SEND NEXT CHARACTER IN SEND BUFFER TO MOST
		=1802 ;	CHECK FOR AND RECEIVE COMMAND FROM MOST (SUPER-8).
		=1803 ;	ENTRY: R80 SELECTED
		=1804 ;	EXIT: USES R0, R1, R2, R3, A
		=1805 ;	
50	0900	=1806	NDMM EQU \$
	0900 0A	=1807	IN A, P2
	0901 F211	=1808	J87 NDMM05 ;JUMP IF SUPER-8 DOES NOT HAVE DATA
		=1809 ;	
		=1810 ;	DATA AVAILABLE - READ AND SET HANDSHAKE
		=1811 ;	
55	0903 8808	=1812	MOV R0, #SUP8 ;ADDRESS OF SUPER-8 DATA BYTE

LOC	OBJ	LINE	SOURCE STATEMENT
5	0906 99FB 0908 A8	=1814 =1815 =1816 ;	ANL P1,#255-EP12 ;P12 LOW TO ACKNOWLEDGE RECEIPT OF DATA MOV R0,A ;SAVE BYTE
	0909 0A 090A 37 090B F209 090D 8904	=1817 NCON03: IN A,P2 =1818 CPL A =1819 JBT NCON03 =1820 ORL P1,#EP12	;JUMP WAIT FOR SUPER-8 TO ACKNOWLEDGE ;P12 HIGH TO END HANDSHAKE SEQUENCE
10		=1821 ; =1822 ; =1823 ;	CHECK RECEIVED COMMAND
	090F 245E	=1824 =1825 ; =1826 ; =1827 ;	JMP CCRCV ;GO EXECUTE COMMAND SUPER-8 IN RECEIVE MODE - CHECK FOR DATA AVAILABLE AND SEND
15	0911 FC 0912 721A 0914 37 0915 8250 0917 F409 0919 83	=1828 NCON05: MOV A,R4 =1829 JBT NCON10 =1830 CPL A =1831 JBT NCON90 =1832 CALL BUFMAN =1833 RET	;JUMP IF SEND BUFFER HAS DATA ;JUMP IF BUFMAN REQUEST FLAG NOT SET ;PUT MESSAGE INTO THE COMM BUFFER
	091A 2650	=1834 ; =1835 NCON10: JNTD NCON90 =1836 ;	;JUMP IF MOST NOT READY
20	091C 8966 091E F1 091F 97 0920 67 0921 11 0922 A9	=1837 MOV R1,#SBFPNT =1838 MOV A,R1 =1839 CLR C =1840 RRC A =1841 INC R1 =1842 MOV R1,A =1843 ;	;GET POINTER ADDRESS ;GET POINTER ;PUT NIBBLE POINTER IN CARRY ;INCREMENT POINTER ;BYTE ADDRESS
25	0923 F1 0924 53F0 0926 D3F0 0928 C630	=1844 MOV A,R1 =1845 ANL A,#0F0H =1846 XRL A,#0F0H =1847 JZ NCON50 =1848 ;	;GET DATA ;MASK POSSIBLE TERMINATOR FLAG ;TEST FOR TERMINATOR FLAG, 0F0H ;JUMP IF TERMINATOR FLAG
	092A F1 092B D3CC 092D C653	=1849 MOV A,R1 =1850 XRL A,#ETRMBY =1851 JZ NCON70 =1852 ;	;GET DATA AGAIN ;JUMP IF TERMINATION BYTE
30	092F F1 0930 F633	=1853 MOV A,R1 =1854 JC NCON20 =1855 ;	;GET DATA AGAIN ;JUMP IF LOW NIBBLE IS NEXT
	0932 47 0933 530F 0935 A8 0936 D30C 0938 C650	=1856 SWAP A =1857 NCON20: ANL A,#0F0H =1858 MOV A,R3 =1859 XRL A,#0C0H =1860 JZ NCON90 =1861 ;	;MASK NIBBLE ;SAVE NIBBLE ;JUMP IF FILLER CHARACTER (DON'T SEND)
35	093A F8 093B 2455	=1862 ; IBH-OCR CHARACTER FORMAT =1863 NCON40: MOV A,R3 =1864 JMP NCON80 =1865 ;	;GET CHARACTER
	093D 8866 093F 10 0940 884A	=1866 ; PROCESS THE TERMINATOR FLAG BYTE =1867 NCON50: MOV R0,#SBFPNT =1868 INC R0 =1869 MOV R3,#LOW NCTBL1 =1870 ;	;COMM BUFFER POINTER ADDRESS ;INCREMENT PAST TERMINATION FLAG BYTE ;IBH-OCR TABLE ADDRESS
40	0942 F1 0943 530F 0945 07 0946 68 0947 A3 0948 2455	=1871 NCON60: MOV A,R1 =1872 ANL A,#0F0H =1873 DEC A =1874 ADD A,R3 =1875 MOV A,BA =1876 JMP NCON80 =1877 ;	;GET TERMINATION FLAG BYTE ;MASK VERSION POINTER, 1 TO 9 ;ADJUST POINTER, 0 TO 8 ;VERSION POINTER + TABLE ADDRESS ;GET TERMINATION CHARACTER
45		=1878 ; =1879 NCTBL1 EQU S	;TABLE START ADDRESS
	094A 00 094B 16 094C 0A 094D 0C 094E 00 094F 00 0950 00 0951 00 0952 00	=1880 DB 00H =1881 DB 16H =1882 DB 0AH =1883 DB 0CH =1884 DB 00H =1885 DB 00H =1886 DB 00H =1887 DB 00H =1888 DB 00H =1889 ;	; A ; 13 ; E ; 8 ; D1, NOT DEFINED ; D2, NOT DEFINED ; D3, NOT DEFINED ; D4, NOT DEFINED ; D5, NOT DEFINED

ISIS-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
GMA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 26

```

5      LOC 08J      LINE      SOURCE STATEMENT

      0953 2380      =1891 MCON70: MOV    A,880H      ;EOT WITH DUMMY DATA (OON WITH 87 SET)
                        =1892 ;
                        =1893 ; SEND FORMATED CHARACTER TO SUPER-8
10      0955 8808      =1894 MCON80: MOV    R0,8808
      0957 90          =1895      MOVX    8R0,A      ;OUTPUT FORMATTED CHARACTER
      0958 37          =1896      CPL     A
      0959 F250        =1897      J87     MCON90      ;JUMP IF NOT END OF DATA
      0958 1416        =1898      CALL    CLR8BF      ;CLEAR SEND BUFFER, ETC.
      0950 83          =1899 MCON90: RET
                        =1900 ;
                        =1901 ; ROUTINE: CKRCV
15      095E          =1902 ; FUNCTION: CHECK RECEIVED BYTE COMMAND AND EXECUTE IT
      095E F8          =1903 ; ENTRY: R0 IS RECEIVED BYTE
                        =1904 ; EXIT:  COMMAND EXECUTED
                        =1905 ;
      095E          =1906 CKRCV  EQU     S
      095E F8          =1907      MOV     A,R0
                        =1908 ;
20      095F 0311      =1909 CKRCV1: XRL    A,#ENSCAN
      0961 9666        =1910      JNZ     CKRCV2      ;JUMP IF NOT ENABLE SCANNING COMMAND
      0963 99EF        =1911      ANL     P1,80FFH-ELASDB ;TURN ON LASER
      0965 83          =1912      RET
                        =1913 ;
      0966 F8          =1914 CKRCV2: MOV    A,R0
      0967 0312        =1915      XRL     A,#DISCAN
      0969 966E        =1916      JNZ     CKRCV3      ;JUMP IF NOT DISABLE SCANNING COMMAND
25      0968 8910      =1917      ORL     P1,8ELASDB   ;TURN OFF LASER
      0960 83          =1918      RET
                        =1919 ;
      096E F8          =1920 CKRCV3: MOV    A,R0
      096F 0314        =1921      XRL     A,#ENBEEP
      0971 9677        =1922      JNZ     CKRCV4      ;JUMP IF NOT ENABLE TONE COMMAND
      0973 05          =1923      SEL     R81
      0974 A8          =1924      MOV     R3,A      ;SET TONE ENABLE FLAG TO 0 (ENABLE TONE)
      0975 C5          =1925      SEL     R80
30      0976 83          =1926      RET
                        =1927 ;
      0977 F8          =1928 CKRCV4: MOV    A,R0
      0978 0318        =1929      XRL     A,#IBEEP
      097A 9681        =1930      JNZ     CKRCV5      ;JUMP IF NOT DISABLE TONE COMMAND
      097C 37          =1931      CPL     A
      097D 05          =1932      SEL     R81
      097E A8          =1933      MOV     R3,A      ;SET TONE ENABLE FLAG TO 1'S (DISABLE TONE)
35      097F C5          =1934      SEL     R80
      0980 83          =1935      RET
                        =1936 ;
      0981 F8          =1937 CKRCV5: MOV    A,R0
      0982 0332        =1938      XRL     A,#CONRST
      0984 9689        =1939      JNZ     CKRCV6      ;JUMP IF NOT RESET COMMAND
      0986 E5          =1940      SEL     R80
40      0987 0400      =1941      JMP     RSTTRP      ;RESET SCANNER
                        =1942 ;
      0989 F8          =1943 CKRCV6: MOV    A,R0
      098A 0377        =1944      XRL     A,#IFRSG     ; (ADDED 1/21/88)
                        =1945 ;
                        =1946 CKRCV7: RET      ;INVALID COMMAND
      098C 83          =1947 S      INCLUDE(:F1:PROCSG.SRC)
                        =1948 ;
45      098E          =1949 ; FILE: PROCSG.SRC 07-03-86 15:15 BOB ACTIS
                        =1950 ;
                        =1951 ; ROUTINE: SUP4SH, SUP6SH
                        =1952 ; FUNCTION: SWAP (REVERSE) ORDER OF PACKED CHARACTERS IN SCAN BUFFER.
                        =1953 ; SUP4SH SWAPS CHARACTERS IN SCHBUF+1 AND SCHBUF+2.
                        =1954 ; SUP6SH SWAPS CHARACTERS IN SCHBUF, SCHBUF+1 AND SCHBUF+2.
                        =1955 ; CLEARS BACKWARD BIT IN THE PARITY DECODE BYTE SCHBUF+3.
60      098E          =1956 ; ENTRY: NO SETUP
                        =1957 ; EXIT: CHARACTERS SWAPPED.
                        =1958 ; R0 = SCHBUF+3 (PARITY DECODE BYTE ADDRESS)
                        =1959 ; BACKWARD BIT IS CLEARED.
                        =1960 ; A = PARITY DECODE BYTE
                        =1961 ;
      098D 8822        =1962 SUP4SH: MOV    R0,#SCHBUF+2
      098F F0          =1963      MOV     A,R0
      0990 47          =1964      SWAP    A
      0991 C8          =1965      DEC     R0
55      0992 20          =1966      XCH     A,R0

```

5

1515-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
GHA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 27

	LOC	OBJ	LINE	SOURCE STATEMENT
	0994	18	=1968	INC RO
	0995	A0	=1969	MOV BRO,A
	0996	24A6	=1970	JMP SUPXEX
10			=1971 ;	
	0998	B022	=1972	SUP6SH: MOV RO,#SCHBUF+2
	099A	F0	=1973	MOV A,BRO
	099B	47	=1974	SWAP A
	099C	C8	=1975	DEC RO
	099D	20	=1976	XCH A,BRO
	099E	47	=1977	SWAP A
	099F	20	=1978	XCH A,BRO
15			=1979	DEC RO
	09A0	C8	=1980	XCH A,BRO
	09A1	20	=1981	SWAP A
	09A2	47	=1982	INC RO
	09A3	18	=1983	INC RO
	09A4	18	=1984	MOV BRO,A
	09A5	A0	=1985 ;	
	09A6	18	=1986	SUPXEX: INC RO
20			=1987	MOV A,BRO
	09A7	F0	=1988	ANL A,#255-EDECBX
	09A8	538F	=1989	MOV BRO,A
	09AA	A0	=1990	JMP SUPXRT
	09AB	4409	=1991 ;	
			=1992 ;	ROUTINE: INCRNB, INCLNB
			=1993 ;	FUNCTION: INCREMENT SEGMENT COUNTERS.
25			=1994 ;	LOW NIBBLE IS SEGMENT ONE COUNTER.
			=1995 ;	HIGH NIBBLE IS SEGMENT TWO COUNTER.
			=1996 ;	TERMINAL COUNT IS 15. (OFN)
			=1997 ;	IF NOT ALREADY TERMINAL COUNT, INCREMENT THE SEGMENT
			=1998 ;	COUNTER AND TOTAL COUNTER.
			=1999 ;	ENTRY: RO = PACKED SEGMENT COUNTER ADDRESS
			=2000 ;	RO+1 = SEGMENT TOTAL COUNTER ADDRESS
			=2001 ;	EXIT: IF NIBBLE WAS INCREMENTED:
30			=2002 ;	RO = SEGMENT TOTAL COUNTER ADDRESS
			=2003 ;	A = PACKED COUNTER
			=2004 ;	CARRY = CLEAR
			=2005 ;	IF NIBBLE WAS ALREADY OFN:
			=2006 ;	RO = PACKED SEGMENT COUNTER ADDRESS
			=2007 ;	A = NOT DEFINED
			=2008 ;	CARRY = SET
			=2009 ;	
35	09AD	F0	=2010	INCRNB: MOV A,BRO ;GET PACKED COUNTER
	09AE	97	=2011	CLR C
	09AF	0310	=2012	ADD A,#10H ;INCREMENT HIGH NIBBLE
	09B1	F6C0	=2013	JC INCRNT ;JUMP IF ALREADY OFN
	09B3	248D	=2014	JMP INCRNT
			=2015 ;	
	09B5	F0	=2016	INCLNB: MOV A,BRO
	09B6	47	=2017	SWAP A
40			=2018	CLR C
	09B8	0310	=2019	ADD A,#10H
	09BA	F6C0	=2020	JC INCRNT ;JUMP IF ALREADY OFN
	09BC	47	=2021	SWAP A
			=2022 ;	
	09BD	A0	=2023	INCRNT: MOV BRO,A ;UPDATE THE PACKED COUNTER
	09BE	18	=2024	INC RO
45			=2025	INC BRO ;INCREMENT TOTAL COUNTER
	09C0	83	=2026 ;	
			=2027	INCRNT: RET
			=2028 ;	
			=2029 ;	ROUTINE: MCK2BT, MCK3BT, MCK4BT
			=2030 ;	FUNCTION: COMPARE BYTES IN ONE BUFFER WITH A SECOND BUFFER.
			=2031 ;	ENTRY: R0 = FIRST BYTE ADDRESS OF FIRST BUFFER
			=2032 ;	R1 = FIRST BYTE ADDRESS OF SECOND BUFFER
50			=2033 ;	EXIT: IF BUFFER ONE EQUALS BUFFER TWO:
			=2034 ;	R0 = END OF FIRST BUFFER ADDRESS PLUS 1.
			=2035 ;	R1 = END OF SECOND BUFFER ADDRESS PLUS 1.
			=2036 ;	R2 = 0
			=2037 ;	A = 0
			=2038 ;	IF BUFFER ONE DOESN'T EQUAL BUFFER TWO:
			=2039 ;	R0 = BUFFER ONE "NOT EQUAL" BYTE ADDRESS
			=2040 ;	R1 = BUFFER TWO "NOT EQUAL" BYTE ADDRESS
55			=2041 ;	R2 = 0
			=2042 ;	A = 0
			=2043 ;	

IS18-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
 GNA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 28

```

5
LOC OBJ      LINE      SOURCE STATEMENT
09C3 24C8     =2045      JMP      NCIOBY
09C5 8A03     =2046      MOV      R2,#3
09C7 24C8     =2047      JMP      NCIOBY
09C9 8A04     =2048      MOV      R2,#4
10 09CB F0     =2049      MOV      A,R0
09CC 01       =2050      XRL      A,R1
09CD 9603     =2051      JNZ      NCIOBY      ;JUMP IF NOT EQUAL
09CF 18       =2052      INC      R0
09D0 19       =2053      INC      R1
09D1 8AC8     =2054      DJNZ     R2,NCIOBY
09D3 83       =2055      NCIOBY: RET
          =2056      ;-----
15 09D4 A3     =2057      TROPG9: MOV    A,DA
09D5 83       =2058      RET
0A00          =2059      ORG      0A00H
          =2060      ;-----
          =2061      ROUTINE: PROC5G
          =2062      FUNCTION: CHECK FOR SCAN BUFFER DATA.
          =2063      SWAP SCAN BUFFER DATA IF BACKWARDS.
          =2064      CHECK FOR MISMATCHES.
20 09D6 F0     =2065      MOVE SCAN BUFFER TO PROPER SEGMENT BUFFER.
          =2066      INCREMENT SEGMENT AND TOTAL COUNTERS.
          =2067      ENTRY: NO SETUP
          =2068      EXIT: USES R0,R1,R2,R3,A
          =2069      ;
0A00 2480     =2070      SUP4SJ: JMP    SUP4SN
0A02 2498     =2071      SUP6SJ: JMP    SUP6SN
          =2072      ;
25 0A04 8523   =2073      PROC5G: MOV    R0,#SCMBUF+3
0A06 F0       =2074      MOV      A,R0      ;GET PARITY DECODE BYTE
0A07 C650     =2075      JZ       PROCR1    ;JUMP IF NO DATA
          =2076      ;
0A09 8228     =2077      SUPXRT: JBS    PROCDX    ;JUMP IF UPC-D BIT SET
0A0B 9230     =2078      JB4     PROCE     ;JUMP IF UPC-E BIT SET
0A0D 530F     =2079      ANL     A,#0FH
0A0F 0300     =2080      ADD     A,#0      ;SETS CARRY FLAGS FOR DA
0A11 57       =2081      DA       A
0A12 9219     =2082      JB4     PROCD5    ;JUMP IF DECODE > 9
          =2083      ;
0A14          =2084      PROC5L: EQU    $
0A16 F0       =2085      PROC5: MOV    A,R0      ;PROCESS AN EAN-13-L
0A18 D202     =2086      JB6     SUP6SJ    ;PROCESS A D-SEGMENT
0A17 6400     =2087      JMP     PRO7CH    ;JUMP IF BACKWARDS
          =2088      ;
35 0A19 3245   =2089      PROC55: JB1     PROCA     ;JUMP IF UPC-A
0A1B F0       =2090      PROC5: MOV    A,R0      ;EAN-8 COMES HERE
0A1C 0200     =2091      JB6     SUP4SJ    ;JUMP IF BACKWARDS
0A1E 1224     =2092      JB0     PROC6R    ;JUMP IF EAN-8-R
0A20 8B3A     =2093      PROC5L: MOV    R3,#L4SCNT
0A22 4451     =2094      JMP     PROC6H
0A24 8B40     =2095      PROC6R: MOV    R3,#R4SCNT
40 0A26 4451   =2096      JMP     PROC6H
          =2097      ;
0A28 530F     =2098      PROCDX: ANL    A,#0FH
0A2A C614     =2099      JZ       PROCD      ;JUMP IF UPC-D SEG
0A2C F0       =2100      PROCDH: MOV    A,R0
0A2D 0200     =2101      JB6     SUP4SJ
0A2F 530F     =2102      ANL     A,#0FH      ;PROCESS N(1) TO N(6) SEGS
0A31 0336     =2103      ADD     A,#LOW PROCHT-1
45 0A33 A3     =2104      MOV     A,DA      ;GET THE SEGMENT COUNTER ADDRESS
0A34 A8       =2105      MOV     R3,A
0A35 4451     =2106      JMP     PROC6H
0A37 46       =2107      PROCHT: DB     LOW M1SCNT
0A38 4C       =2108      DB     LOW M2SCNT
0A39 52       =2109      DB     LOW M3SCNT
0A3A 58       =2110      DB     LOW M4SCNT
0A3B 5E       =2111      DB     LOW M5SCNT
50 0A3C 64     =2112      DB     LOW M6SCNT
          =2113      ;
0A3D 0202     =2114      PROC6: JB6     SUP6SJ
0A3F 1466     =2115      CALL    ENCD10
0A41 964E     =2116      JNZ     PROC6X      ;JUMP IF ENCD10 TEST FAILED
0A43 6400     =2117      JMP     PRO7CH
          =2118      ;
55 0A45 F0     =2119      PROCA: MOV    A,R0
0A46 0202     =2120      JB6     SUP6SJ

```

ISIS-II MCS-46/UP1-41 MACRO ASSEMBLER, V4.2
QMA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 29

5

10

15

20

25

30

35

40

45

50

55

LOC	OBJ	LINE	SOURCE STATEMENT
0A4A	6400	=2122	PROCAL: JNP PRO7CH
0A4C	4495	=2123	PROCAR: JNP PRO6CH
		=2124	;
0A4E	140C	=2125	PROCEX: CALL CLRSHB ;GO CLEAR THE SCAN BUFFER
0A50	83	=2126	PROCRT: RET ;RETURN FROM PROCESS SEGMENT ROUTINE
		=2127	*****
		=2128	ROUTINE: PRO4CH
		=2129	FUNCTION: PROCESS 4-CHAR SEGMENTS (2 BYTES)
		=2130	CHECKS FOR MISMATCHES.
		=2131	MOVES SEGMENT FROM SCAN BUFFER TO SEGMENT BUFFER.
		=2132	INCREMENTS SCAN AND TOTAL COUNTERS.
		=2133	ENTRY: R3 = PACKED SCAN COUNTER ADDRESS
		=2134	R3-4 = SCAN 1 BUFFER ADDRESS
		=2135	R3-2 = SCAN 2 BUFFER ADDRESS
		=2136	R3+1 = TOTAL COUNTER ADDRESS
		=2137	EXIT: SEGMENT PROCESSED
		=2138	SCAN BUFFER CLEARED
		=2139	USES R0,R1,R2,R3,A
		=2140	;
0A51	FB	=2141	PRO4CH: MOV A,R3
0A52	AB	=2142	MOV R0,A
0A53	FD	=2143	MOV A,R0
			;GET S2/S1 PACKED COUNTS
0A54	530F	=2144	ANL A,#0FH
0A56	8821	=2145	MOV R0,#SCANBUF+1
0A58	9662	=2146	JNZ PRO4C2 ;JUMP IF SCAN 1 COUNTER <= 0
		=2147	;
0A5A	FB	=2148	MOV A,R3
			;SCAN 1 COUNTER = 0
0A5B	03FC	=2149	ADD A,#-4
0A5D	A9	=2150	MOV R1,A
			;SCAN 1 BUFFER ADDRESS
0A5E	1426	=2151	CALL MOV2BY
0A60	446A	=2152	JMP PRO4C3
		=2153	;
0A62	FB	=2154	PRO4C2: MOV A,R3
			;SCAN 1 COUNTER <= 0
0A63	03FC	=2155	ADD A,#-4
0A65	A9	=2156	MOV R1,A
			;SCAN 1 BUFFER ADDRESS
0A66	34C1	=2157	CALL MCK2BY
0A68	9670	=2158	JNZ PRO4C4 ;JUMP IF NO MATCH
		=2159	;
0A6A	FB	=2160	PRO4C3: MOV A,R3
0A6B	AB	=2161	MOV R0,A
0A6C	34B5	=2162	CALL INCLNB
			;INCREMENT SCAN 1 COUNTER AND TOTAL
0A6E	444E	=2163	JMP PROCEX
		=2164	;
0A70	FB	=2165	PRO4C4: MOV A,R3
			;SCAN 1 BUFFER DOESN'T MATCH
0A71	AB	=2166	MOV R0,A
0A72	FD	=2167	MOV A,R0
0A73	53FD	=2168	ANL A,#0FDH
0A75	8821	=2169	MOV R0,#SCANBUF+1
0A77	9681	=2170	JNZ PRO4C6 ;JUMP IF SCAN 2 COUNTER <= 0
		=2171	;
0A79	FB	=2172	MOV A,R3
0A7A	03FE	=2173	ADD A,#-2
0A7C	A9	=2174	MOV R1,A
			;SCAN 2 BUFFER ADDRESS
0A7D	1426	=2175	CALL MOV2BY
0A7F	44B9	=2176	JMP PRO4C7
		=2177	;
0A81	FB	=2178	PRO4C6: MOV A,R3
			;SCAN 2 COUNT <= 0
0A82	03FE	=2179	ADD A,#-2
0A84	A9	=2180	MOV R1,A
			;SCAN 2 BUFFER ADDRESS
0A85	34C1	=2181	CALL MCK2BY
0A87	968F	=2182	JNZ PRO4C8 ;JUMP IF NO MATCH
		=2183	;
0A89	FB	=2184	PRO4C7: MOV A,R3
0A8A	AB	=2185	MOV R0,A
0A8B	34AD	=2186	CALL INCLNB
			;INCREMENT SCAN 2 COUNT AND TOTAL
0A8D	444E	=2187	JMP PROCEX
		=2188	;
0A8F	FB	=2189	PRO4C8: MOV A,R3
			;NEITHER SCAN BUFFER MATCHED
0A90	17	=2190	INC A
0A91	AB	=2191	MOV R0,A
			;TOTAL COUNTER ADDRESS
0A92	10	=2192	INC R0
			;INCREMENT TOTAL COUNTER
0A93	444E	=2193	JMP PROCEX
		=2194	*****
		=2195	ROUTINE: PRO6CH
		=2196	FUNCTION: PROCESS 6-CHAR SEGMENTS (3 BYTES)
		=2197	CHECKS FOR MISMATCHES.

ISIS-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
 QAD95 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 30

LOC	OBJ	LINE	SOURCE STATEMENT
		=2199 ;	INCREMENTS SCAN AND TOTAL COUNTERS.
		=2200 ;	ENTRY: NO SETUP
		=2201 ;	EXIT: SEGMENT IS PROCESSED
		=2202 ;	SCAN BUFFER CLEARED
		=2203 ;	USES R0,R1,R2,R3,A
		=2204 ;	
	0A95 8834	=2205 PRO6CH: MOV	R0,#R6SCHT
	0A97 F0	=2206 MOV	A,BRO
	0A98 530F	=2207 ANL	A,#0F0H
	0A9A 892E	=2208 MOV	R1,#R6S1
	0A9C 8820	=2209 MOV	R0,#SCHBUF
	0A9E 96A4	=2210 JNZ	PRO6C2
		=2211 ;	;JUMP IF SCAN 1 COUNTER <= 0
	0AA0 142A	=2212 CALL	MOV3BY
	0AA2 44A8	=2213 JNP	PRO6C3
		=2214 ;	
	0AA4 34C5	=2215 PRO6C2: CALL	MOV3BY
	0AA6 96AE	=2216 JNZ	PRO6C4
		=2217 ;	;SCAN 1 COUNTER <= 0
			;JUMP IF NO MATCH
	0AA8 8834	=2218 PRO6C3: MOV	R0,#R6SCHT
	0AAA 3485	=2219 CALL	INCLNB
	0AAC 444E	=2220 JNP	PROCEX
		=2221 ;	;INCREMENT SCAN 1 COUNTER AND TOTAL
	0AAE 8834	=2222 PRO6C4: MOV	R0,#R6SCHT
	0AB0 F0	=2223 MOV	A,BRO
	0AB1 53F0	=2224 ANL	A,#0F0H
	0AB3 8931	=2225 MOV	R1,#R6S2
	0AB5 8820	=2226 MOV	R0,#SCHBUF
	0AB7 9680	=2227 JNZ	PRO6C6
		=2228 ;	;JUMP IF SCAN 2 COUNTER <= 0
	0AB9 142A	=2229 CALL	MOV3BY
	0ABB 44C1	=2230 JNP	PRO6C7
		=2231 ;	
	0ABD 34C5	=2232 PRO6C6: CALL	MOV3BY
	0ABF 96C7	=2233 JNZ	PRO6C8
		=2234 ;	;SCAN 2 COUNTER <= 0
			;JUMP IF NO MATCH
	0AC1 8834	=2235 PRO6C7: MOV	R0,#R6SCHT
	0AC3 34AD	=2236 CALL	INCLNB
	0AC5 444E	=2237 JNP	PROCEX
		=2238 ;	;INCREMENT SCAN 2 COUNT AND TOTAL
	0AC7 8835	=2239 PRO6C8: MOV	R0,#R6STOT
	0AC9 10	=2240 INC	BRO
	0ACA 444E	=2241 JNP	PROCEX
		=2242 ;	*****
	0ACC A3	=2243 TROPGA: MOV	A,BR
	0ACD 83	=2244 RET	
	0AD0	=2245 ORG	0800H
		=2246 ;	*****
		=2247 ;	ROUTINE: PRO7CH
		=2248 ;	FUNCTION: PROCESS 7-CHAR SEGMENTS (4 BYTES)
		=2249 ;	CHECKS FOR MISMATCHES.
		=2250 ;	MOVES SEGMENT FROM SCAN BUFFER TO SEGMENT BUFFER.
		=2251 ;	INCREMENTS SCAN AND TOTAL COUNTERS.
		=2252 ;	ENTRY: NO SETUP
		=2253 ;	EXIT: SEGMENT IS PROCESSED
		=2254 ;	SCAN BUFFER CLEARED
		=2255 ;	USES R0,R1,R2,R3,A
		=2256 ;	
	0B00 882C	=2257 PRO7CH: MOV	R0,#R6SCHT
	0B02 F0	=2258 MOV	A,BRO
	0B03 530F	=2259 ANL	A,#0F0H
	0B05 8924	=2260 MOV	R1,#R6S1
	0B07 8820	=2261 MOV	R0,#SCHBUF
	0B09 960F	=2262 JNZ	PRO7C2
		=2263 ;	;JUMP IF SCAN 1 COUNTER <= 0
	0B0B 142E	=2264 CALL	MOV4BY
	0B0D 6413	=2265 JNP	PRO7C3
		=2266 ;	;SCAN 1 COUNTER = 0
	0B0F 34C9	=2267 PRO7C2: CALL	MOV4BY
	0B11 9619	=2268 JNZ	PRO7C4
		=2269 ;	;JUMP IF NO MATCH
	0B13 882C	=2270 PRO7C3: MOV	R0,#R6SCHT
	0B15 3485	=2271 CALL	INCLNB
	0B17 444E	=2272 JNP	PROCEX
		=2273 ;	;INCREMENT SCAN 1 COUNTER AND TOTAL
	0B19 882C	=2274 PRO7C4: MOV	R0,#R6SCHT
			;SCAN 1 BUFFER DOESN'T MATCH

ISIS-II MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
QMA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 31

```

5      LOC OBJ      LINE      SOURCE STATEMENT
      081C 53F0      =2276      ANL      A,#0F0H
      081E 8928      =2277      MOV      R1,#1652
      0820 8820      =2278      MOV      R0,#SCNBUF
      0822 9628      =2279      JNZ      PRO7C6      ;JUMP IF SCAN 2 COUNTER < 0
      0824 142E      =2280 ;
      0826 642C      =2281      CALL     MOV4BY
      10      =2282      JMP      PRO7C7
      =2283 ;
      0828 34C9      =2284 PRO7C6: CALL   MCN4BY      ;SCAN 2 COUNTER < 0
      082A 9632      =2285      JNZ      PRO7C8      ;JUMP IF NO MATCH
      =2286 ;
      082C 882C      =2287 PRO7C7: MOV    R0,#165CNT
      082E 34AD      =2288      CALL     INCRNB
      15      0830 444E      =2289      JMP      PROCEX      ;INCREMENT SCAN 2 COUNT AND TOTAL
      =2290 ;
      0832 832D      =2291 PRO7C8: MOV    R0,#165TOT
      0834 10        =2292      INC      DRO
      0835 444E      =2293      JMP      PROCEX      ;INCREMENT TOTAL COUNTER
      =2294 ;
      =2295      INCLUDE(=F1:VERTAG.SRC)
      20      =2296 ; FILE: VERTAG.SRC 12-09-86 13:50 BOB ACTIS
      =2297 ;
      =2298 ; ROUTINE: CCMJ
      =2299 ; FUNCTION: DETERMINE MAJORITY SEGMENT COUNTER
      =2300 ; ENTRY: R0 = PACKED COUNTER ADDRESS
      =2301 ; EXIT: R0 = NOT CHANGED
      =2302 ; USES R2,A
      =2303 ; CARRY SET IF LOW-HIGH NIBBLE (S1CNT>S2CNT)
      25      =2304 ; CARRY CLEAR IF LOW-HIGH NIBBLE (S1CNT<S2CNT)
      =2305 ;
      0837 F0        =2306 CCMJ: MOV     A,DRO      ;GET PACKED COUNTER
      0838 530F      =2307      ANL      A,#0F0H
      083A AA        =2308      MOV      R2,A
      083B F0        =2309      MOV      A,DRO      ;S1CNT
      083C 47        =2310      SWAP     A
      083D 530F      =2311      ANL      A,#0F0H
      083F C645      =2312      JZ       CCMJ9
      30      0841 37      =2313      CPL      A
      0842 17        =2314      INC      A
      0843 6A        =2315      ADD      A,R2
      0844 83        =2316      RET
      =2317 ;
      0845 97        =2318 CCMJ9: CLR     C
      0846 A7        =2319      CPL      C
      35      0847 83      =2320      RET
      =2321 ;
      =2322 ; ROUTINE: CCMJSH
      =2323 ; FUNCTION: CHECK FOR EXCESS MISMATCHED SEGMENTS.
      =2324 ; CLEAR COUNTERS IF EXCESS MISMATCHES.
      =2325 ; ENTRY: SCAN 1 COUNTER HAS THE MAJORITY SEGMENT COUNT.
      =2326 ; R0 = PACKED SCAN2/SCAN1 SEGMENT COUNTER ADDRESS.
      40      =2327 ; R0+1 = TOTAL SEGMENT COUNTER ADDRESS.
      =2328 ; EXIT: USES R1,R2,A
      =2329 ; R0 = NOT CHANGED
      =2330 ; IF TOTAL=MAJORITY (0 MISMATCHES)
      =2331 ; OR TOTAL-1=MAJORITY (1 MISMATCH) AND MAJORITY>=3
      =2332 ; OR TOTAL-2=MAJORITY (2 MISMATCH) AND MAJORITY>=15
      =2333 ; THEN RETURN,
      =2334 ; ELSE COUNTERS ARE CLEARED.
      =2335 ;
      45      0848 F0      =2336 CCMJSH: MOV    A,DRO
      0849 C66F      =2337      JZ      CCMJS90      ;JUMP IF S2/S1 COUNTERS=0
      =2338 ;
      084B F8        =2339      MOV      A,R0
      084C 17        =2340      INC      A
      084D A9        =2341      MOV      R1,A
      =2342 ;
      50      084E F0      =2343      MOV      A,DRO
      084F 530F      =2344      ANL      A,#0F0H
      0851 AA        =2345      MOV      R2,A
      0852 D1        =2346      XRL      A,DRO
      0853 C66F      =2347      JZ      CCMJS90
      =2348 ;
      0855 F1        =2349      MOV      A,DRO
      0856 07        =2350      DEC      A
      55      0857 DA      =2351      XRL      A,R2

```

1515-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
 QUAOPS ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 32

5

10

15

20

25

30

35

40

45

50

55

```

LOC OBJ      LINE      SOURCE STATEMENT

      =2353 ;
085A F1      =2354      MOV     A,R1
085B 07      =2355      DEC     A
085C 07      =2356      DEC     A
085D DA      =2357      XRL     A,R2
085E 966C    =2358      JNZ     CIMS80      ;JUMP IF >2 MISMATCHES
      =2359 ;
      =2360 ;      TOTAL-2=MAJORITY (2 MISMATCHES).  CHECK FOR MAJORITY>=15.
      =2361 ;
0860 FA      =2362      MOV     A,R2
0861 03F1    =2363      ADD     A,#-15
0863 E66C    =2364      JNC     CIMS80      ;JUMP IF <15
0865 83      =2365      RET
      =2366 ;
      =2367 ;      TOTAL-1=MAJORITY (1 MISMATCH).  CHECK FOR MAJORITY>=3.
      =2368 ;
0866 FA      =2369      CIMS30: MOV    A,R2
0867 03FD    =2370      ADD     A,#-3
0869 E66C    =2371      JNC     CIMS80      ;JUMP IF <3
0868 83      =2372      RET
      =2373 ;
      =2374 ;      FAILED TEST.  CLEAR COUNTERS.
      =2375 ;
086C 27      =2376      CIMS80: CLR    A
086D A0      =2377      MOV     DR0,A      ;CLEAR S2/S1 COUNTERS
086E A1      =2378      MOV     DR1,A      ;CLEAR TOTAL COUNTER
086F 83      =2379      CIMS90: RET
      =2380 ;
      =2381 ;      ROUTINE: EXS8F2, EXS8F3, EXS8F4
      =2382 ;      FUNCTION:  EXCHANGE SEGMENT BUFFERS AND COUNTERS. (2,3 OR 4 BYTES)
      =2383 ;      (I.E. EXCHANGE BUFFER 1 DATA WITH BUFFER 2 DATA)
      =2384 ;      ENTRY:  R0 = SCAN 1 BUFFER ADDRESS
      =2385 ;      R0+2(3,4) = SCAN 2 BUFFER ADDRESS
      =2386 ;      R0+3(4,5) = SCAN2/SCAN1 PACKED COUNTER ADDRESS
      =2387 ;      EXIT:  USES R1,R2
      =2388 ;      SCAN 1 AND SCAN 2 DATA AND COUNTERS EXCHANGED.
      =2389 ;      R0 = SCAN2/SCAN1 COUNTER ADDRESS.
      =2390 ;      A = SCAN2/SCAN1 COUNTERS
      =2391 ;
0870 BA02    =2392      EXS8F2: MOV    R2,#2
0872 647A    =2393      JNP     EXS8FX
0874 BA03    =2394      EXS8F3: MOV    R2,#3
0876 647A    =2395      JNP     EXS8FX
0878 BA04    =2396      EXS8F4: MOV    R2,#4
087A F8      =2397      EXS8FX: MOV    A,R0      ;GET SCAN 1 BUFFER ADDRESS
087B A9      =2398      MOV     R1,A      ;SAVE IT
087C 6A      =2399      ADD     A,R2      ;CALCULATE SCAN 2 BUFFER ADDRESS
087D A8      =2400      MOV     R0,A      ;SAVE IT
      =2401 ;
087E F0      =2402      EXS8F1: MOV    A,DR0      ;GET SCAN 2 DATA
087F 21      =2403      XCH     A,DR1      ;EXCHANGE DATA
0880 A0      =2404      MOV     DR0,A      ;STORE SCAN 1 DATA
0881 18      =2405      INC     R0
0882 19      =2406      INC     R1
0883 EA7E    =2407      DJNZ    R2,EXS8F1
      =2408 ;
0885 F0      =2409      MOV     A,DR0      ;GET S2/S1 COUNTERS
0886 47      =2410      SWAP    A      ;EXCHANGE COUNTERS
0887 A0      =2411      MOV     DR0,A      ;SAVE COUNTERS
0888 83      =2412      RET
      =2413 ;
      =2414 ;      ROUTINE: SUM12C
      =2415 ;      FUNCTION:  CALCULATE L6S1 + R6S1 MOD-10 CHECKSUM.
      =2416 ;      ENTRY:  L6S1 AND R6S1 HAVE DATA TO USE.
      =2417 ;      EXIT:  A = MOD-10 CHECKSUM CALCULATION FOR 12 CHARACTERS.
      =2418 ;      USES R0,R2,R3
      =2419 ;
0889 8824    =2420      SUM12C: MOV    R0,R6S1
088B 1461    =2421      CALL    MOD106
088D A8      =2422      MOV     R3,A      ;SAVE LEFT HALF SUM
      =2423 ;
088E 882E    =2424      MOV     R0,R6S1
0890 1461    =2425      CALL    MOD106
0892 68      =2426      ADD     A,R3      ;RIGHT SUM + LEFT SUM
0893 57      =2427      DA      A
0894 53DF    =2428      ANL     A,#0FH      ;MASK SUM DIGIT

```

ISIS-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
GHA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 33

5

10

15

20

25

30

35

40

45

50

55

```

LOC OBJ      LINE      SOURCE STATEMENT
=====
=2430 ;*****
=2431 ; ROUTINE: CK6TOT
=2432 ; FUNCTION: CHECK L6STOT AND R6STOT FOR COUNTS.
=2433 ; ENTRY: R1 SETUP WITH MINUS THE MINIMUM NUMBER OF SCANS REQUIRED
=2434 ; EXIT: A=0 IF EITHER COUNTER < MINIMUM REQUIRED.
=2435 ; A<0 IF BOTH COUNTERS >= MINIMUM REQUIRED.
=2436 ; USES R0
=2437 ;
0897 B82D    =2438 CK6TOT: MOV    R0,R16STOT
0899 F0      =2439          MOV    A,R0                      ;GET LEFT SEGMENT TOTAL
089A 69      =2440          ADD    A,R1
089B F69F    =2441          JC     CK6T20                      ;JUMP IF L6STOT >= -R1
089D 27      =2442 CK6T10: CLR    A
089E 83      =2443          RET                                ;EITHER L6 OR R6 HAD < MINIMUM
=2444 ;
089F B835    =2445 CK6T20: MOV    R0,R16STOT
08A1 F0      =2446          MOV    A,R0                      ;GET RIGHT SEGMENT TOTAL
08A2 69      =2447          ADD    A,R1
08A3 E69D    =2448          JNC    CK6T10                      ;JUMP R6STOT < -R1
08A5 27      =2449          CLR    A
08A6 37      =2450          CPL    A
08A7 83      =2451          RET                                ;BOTH L6 AND R6 HAD >= MINIMUM
=2452 ;*****
=2453 ; ROUTINE: MAJSGS
=2454 ; FUNCTION: DETERMINE MAJORITY SEGMENTS.
=2455 ; MOVE MAJORITY SEGMENT TO BUFFER #1 IF NECESSARY.
=2456 ; (I.E. EXCHANGE BUFFER 1 AND BUFFER 2 DATA AND COUNTERS.)
=2457 ; CHECK FOR EXCESS MISMATCHES.
=2458 ; IF EXCESS MISMATCHES, CLEAR COUNTERS.
=2459 ; ENTRY: NO SETUP
=2460 ; EXIT: IF THE MISMATCH RATIO IS OK,
=2461 ; SEGMENT BUFFER 1 AND COUNTER 1 HAS MAJORITY.
=2462 ; SEGMENT BUFFER 2 AND COUNTER 2 HAS MINORITY.
=2463 ; IF EXCESS MISMATCHES, SEGMENT COUNTERS ARE CLEARED.
=2464 ;
08A8 B82C    =2465 MAJSGS: MOV    R0,R16SCHT
08AA 7437    =2466          CALL    C0NAJ
08AC F682    =2467          JC     MAJSG0                      ;JUMP IF S1 IS MAJORITY
08AE 8824    =2468          MOV    R0,R16S1
08B0 7478    =2469          CALL    EXS8F4                      ;EXCHANGE S2/S1 DATA AND COUNTERS
08B2 7448    =2470 MAJSG0: CALL    C0N1SH
=2471 ;
08B4 B834    =2472          MOV    R0,R16SCHT
08B6 7437    =2473          CALL    C0NAJ
08B8 F68E    =2474          JC     MAJSG1
08BA 882E    =2475          MOV    R0,R16S1
08BC 7474    =2476          CALL    EXS8F3
08BE 7448    =2477 MAJSG1: CALL    C0N1SH
=2478 ;
08C0 8808    =2479          MOV    R3,R8                      ;NUMBER OF 4-CHAR SEGMENT BUFFERS
08C2 B83A    =2480          MOV    R0,R14SCHT                  ;FIRST S2/S1 COUNTER ADDRESS
=2481 ;
08C4 7437    =2482 MAJSG4: CALL    C0NAJ
08C6 F6CE    =2483          JC     MAJSG5                      ;JUMP IF S1 IS THE MAJORITY
08C8 F8      =2484          MOV    A,R0
08C9 03FC    =2485          ADD    A,#-4                      ;CALCULATE THE S1 DATA BUFFER ADDRESS
08CB A8      =2486          MOV    R0,A
08CC 747D    =2487          CALL    EXS8F2                      ;EXCHANGE S2/S1 DATA AND COUNTS
=2488 ;
08CE 7448    =2489 MAJSG5: CALL    C0N1SH                      ;CHECK MISMATCHES
08D0 F8      =2490          MOV    A,R0
08D1 0306    =2491          ADD    A,R6                      ;CALCULATE NEXT S2/S1 COUNTER ADDRESS
08D3-A8      =2492          MOV    R0,A
08D4 E8C4    =2493          DJNZ   R3,MAJSG4
08D6 83      =2494          RET
=2495 ;*****
08D7 A3      =2496 TROPGB: MOV    A,B4
08D8 83      =2497          RET
0C00         =2498          ORG     0C00H
=2499 ;*****
=2500 ; ROUTINE: VERTAG
=2501 ; FUNCTION: PERFORM MISMATCH TEST AND GET MAJORITY SEGMENT AND COUNT
=2502 ; INTO BUFFER AND COUNTER #1. (MAJSGS)
=2503 ; TRY TO BUILD BLOCKS INTO VERSIONS. (TBLOCK)
=2504 ; ENTRY: NO SETUP
=2505 ; EXIT: A=0 IF ENOUGH BLOCKS FOR A VERSION ARE FOUND.

```


ISIS-11 MCS-45/UP1-41 MACRO ASSEMBLER, V4.2
GNA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 34

5

LOC	OBJ	LINE	SOURCE STATEMENT
		=2507 ;	R6 SET TO INDICATE VALID VERSION IF ONE WAS FOUND.
		=2508 ;	R1 SETUP WITH MINUS THE MINIMUM NUMBER OF SCANS REQUIRED.
		=2509 ;	
10	0C00 74A8	=2510 VERTAG: CALL	MAJSGS ;MAJORITY/MISMATCH SEGMENT TEST
	0C02 3400	=2511 CALL	NOOPM ;CHECK COIN. MAJSGS TAKES A WHILE.
	0C04 FE	=2512 MOV	A,R6
	0C05 53F0	=2513 ANL	A,#0F0H ;CLEAR THE VERSION POINTER/FLAG.
	0C07 AE	=2514 MOV	R6,A
		=2515 ;	
	0C08 89FE	=2516 MOV	R1,#-2 ;REQUIRE 2 SCANS MINIMUM
	0C0A 887D	=2517 MOV	R0,#CONFIG
	0C0C F0	=2518 MOV	A,R0 ;GET THE CONFIGURATION BYTE
15	0C0D 3211	=2519 TRYBLK	J81 ;JUMP IF 6 CHAR 2 SCAN BIT IS SET
	0C0F 89FF	=2520 MOV	R1,#-1 ;REQUIRE 1 SCAN MINIMUM
		=2521 ;	
	0C11 9463	=2522 TRYBLK: CALL	TBLKA
	0C13 C661	=2523 JZ	VERA ;JUMP IF A GOOD UPC-A BLOCK
		=2524 ;	
	0C15 9480	=2525 CALL	TBLK13
20	0C17 C660	=2526 JZ	VERT3 ;JUMP IF A GOOD EAN-13 BLOCK
		=2527 ;	
	0C19 949F	=2528 CALL	TBLK2
	0C18 C635	=2529 JZ	VERT10 ;JUMP IF A GOOD BLK-2
		=2530 ;	
	0C1D 89FE	=2531 MOV	R1,#-2 ;ALWAYS REQUIRE 2 SCANS MINIMUM FOR E
	0C1F 94AF	=2532 CALL	TBLKE
	0C21 C65F	=2533 JZ	VERE ;JUMP IF A GOOD UPC-E BLOCK
25		=2534 ;	
	0C23 89FE	=2535 MOV	R1,#-2
	0C25 887D	=2536 MOV	R0,#CONFIG
	0C27 F0	=2537 MOV	A,R0 ;GET THE CONFIGURATION BYTE
	0C28 122C	=2538 J80	VERT05 ;JUMP IF 4 CHAR 2 SCAN BIT IS SET
	0C2A 89FF	=2539 MOV	R1,#-1
		=2540 ;	
	0C2C 842F	=2541 VERT05: CALL	TBLK1
30	0C2E C658	=2542 JZ	VERD1 ;JUMP IF A GOOD BLK-1. (D-1)
		=2543 ;	
	0C30 94C2	=2544 CALL	TBLK8
	0C32 C65E	=2545 JZ	VER8 ;JUMP IF A GOOD EAN-8 BLOCK
	0C34 83	=2546 RET	;RETURN IF NOT ENOUGH BLOCKS
		=2547 ;	
	0C35 89FE	=2548 VERT10: MOV	R1,#-2
35	0C37 887D	=2549 MOV	R0,#CONFIG
	0C39 F0	=2550 MOV	A,R0 ;GET THE CONFIGURATION BYTE
	0C3A 123E	=2551 J80	VERT15 ;JUMP IF 4 CHAR 2 SCAN BIT IS SET
	0C3C 89FF	=2552 MOV	R1,#-1
		=2553 ;	
	0C3E 849E	=2554 VERT15: CALL	TBLK5
	0C40 C648	=2555 JZ	VERT20 ;JUMP IF A GOOD BLK-5
		=2556 ;	
	0C42 8406	=2557 CALL	TBLK6
40	0C44 C656	=2558 JZ	VERD3 ;JUMP IF A GOOD BLK-6. (D-3)
		=2559 ;	
	0C46 8466	=2560 CALL	TBLK3
	0C48 C657	=2561 JZ	VERD2 ;JUMP IF A GOOD BLK-3. (D-2)
	0C4A 83	=2562 RET	;RETURN IF NOT ENOUGH BLOCKS
		=2563 ;	
	0C4B 848A	=2564 VERT20: CALL	TBLK7
45	0C4D C654	=2565 JZ	VERD5 ;JUMP IF A GOOD BLK-7. (D-5)
		=2566 ;	
	0C4F 8482	=2567 CALL	TBLK4
	0C51 C655	=2568 JZ	VERD4 ;JUMP IF A GOOD BLK-4. (D-4)
	0C53 83	=2569 RET	;RETURN IF NOT ENOUGH BLOCKS
		=2570 ;	
	0C54 1E	=2571 VERD5: INC	R6 ;SET R6=9
	0C55 1E	=2572 VERD4: INC	R6 ;SET R6=8
50	0C56 1E	=2573 VERD3: INC	R6 ;SET R6=7
	0C57 1E	=2574 VERD2: INC	R6 ;SET R6=6
	0C58 1E	=2575 VERD1: INC	R6 ;SET R6=5
		=2576 ;	
		=2577 ;	
	0C59 2301	=2578 MOV	A,#1 ;NO VERSION 0 ALLOWED
		=2579 ;	
	0C5B 8E00	=2580 VERT80: MOV	R6,#0 ;CLEAR THE VERSION FLAG
65	0C5D 83	=2581 RET	;RETURN W/ A<0, NO VERSIONS
		=2582 ;	

1515-11 MCS-48/UPI-41 MACRO ASSEMBLER, V4.2 PAGE 35
QMA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

LOC	OBJ	LINE	SOURCE STATEMENT
5			
	OC5F 1E	=2584	VERE: INC R6 ;SET R6=3
	OC60 1E	=2585	VER13: INC R6 ;SET R6=2
	OC61 1E	=2586	VERA: INC R6 ;SET R6=1
		=2587	;
10	OC62 83	=2588	VERT90: RET
		=2589	S INCLUDE(:F1:TBLOCK.SRC)
		=2590	;
		=2591	FILE: TBLOCK.SRC 09-11-86 09:10 BOB ACTIS
		=2592	;
		=2593	ROUTINE: TBKA
		=2594	FUNCTION: TRY FOR A VALID UPC-A BLOCK.
15		=2595	CHECK THAT L6 AND R6 HAVE ENOUGH DATA.
		=2596	CHECK THAT L6 IS AN A-L.
		=2597	IF OK SO FAR, CALCULATE MOD-10 CHECK CHARACTER.
		=2598	IF STILL OK, RETURN WITH A=0.
		=2599	IF MOD-10 ERROR, CLEAR 466-CHAR SEGMENT COUNTERS AND
		=2600	CLEAR VERSION POINTER/FLAG.
		=2601	ENTRY: SCAN 1 BUFFER IS MAJORITY SCAN.
		=2602	R1 SETUP WITH MINUS THE MINIMUM NUMBER OF SCANS REQUIRED
20		=2603	EXIT: USES R0,R2,R3,A
		=2604	A=0 IF GOOD BLOCK
		=2605	A<0 IF NO BLOCK
		=2606	;
	OC63 7497	=2607	TBKA: CALL CK6TOT
	OC65 C67D	=2608	JZ TBNONE ;JUMP IF NO LEFT OR NO RIGHT SEGMENTS
		=2609	;
	OC67 B827	=2610	MOV R0,#L6S1+3
25	OC69 F0	=2611	MOV A,BRO ;GET PARITY DECODE BYTE
	OC6A 530C	=2612	ANL A,#00CH
	OC6C 030C	=2613	XRL A,#00CH
	OC6E 967D	=2614	JNZ TBNONE ;JUMP IF NOT AND AL
		=2615	;
	OC70 7489	=2616	CALL SUM12C
	OC72 9675	=2617	JNZ TBERR6 ;GO CALCULATE LEFT + RIGHT CHECKSUM
	OC74 83	=2618	RET ;JUMP IF MOD-10 IS BAD
30		=2619	;
		=2620	THE FOLLOWING IS USED BY OTHER TBLOCK ROUTINES, BUFMAN, & ROTAG
		=2621	;
	OC75	=2622	CLRVER EQU S ;ENTRY POINT TO CLEAR VERSION FLAGS & DATA
	OC75 1400	=2623	TBERR6: CALL CLR6SG ;CLEAR 6-CHAR SEGMENTS AND COUNTERS
	OC77 1406	=2624	TBERR4: CALL CLR4SG ;CLEAR 4-CHAR SEGMENTS AND COUNTERS
	OC79 FE	=2625	MOV A,R6
	OC7A 53F0	=2626	ANL A,#00FH ;CLEAR VERSION POINTER/FLAG
35	OC7C AE	=2627	MOV R6,A
	OC7D 27	=2628	TBNONE: CLR A
	OC7E 37	=2629	CPL A
	OC7F 83	=2630	RET
		=2631	;
		=2632	ROUTINE: TBK13
		=2633	FUNCTION: TRY FOR A VALID EAN-13 BLOCK.
40		=2634	CHECK THAT L6 AND R6 HAVE ENOUGH DATA.
		=2635	CHECK THAT L6 IS AN EAN-13-L.
		=2636	IF OK SO FAR, CALCULATE MOD-10 CHECK CHARACTER.
		=2637	IF STILL OK, RETURN WITH A=0.
		=2638	IF MOD-10 FAILS, CLEAR 466-CHAR SEGMENT COUNTER AND
		=2639	CLEAR VERSION POINTER/FLAG.
		=2640	ENTRY: SCAN 1 BUFFER IS MAJORITY SCAN.
		=2641	R1 SETUP WITH MINUS THE MINIMUM NUMBER OF SCANS REQUIRED
45		=2642	EXIT: USES R0,R2,R3,A
		=2643	A=0 IF GOOD BLOCK.
		=2644	A<0 IF NO BLOCK FOUND.
		=2645	;
	OC80 7497	=2646	TBK13: CALL CK6TOT
	OC82 C67D	=2647	JZ TBNONE ;JUMP IF NO LEFT OR NO RIGHT SEGMENTS
		=2648	;
	OC84 B827	=2649	MOV R0,#L6S1+3
50	OC86 F0	=2650	MOV A,BRO ;GET PARITY DECODE BYTE
	OC87 5330	=2651	ANL A,#DECE+EDEC ;MASK D AND E SEG BITS
	OC89 967D	=2652	JNZ TBNONE ;JUMP IF D OR E SEGMENT
		=2653	;
	OC8B F0	=2654	MOV A,BRO
	OC8C 530F	=2655	ANL A,#00FH ;MASK THE DECODED CHARACTER
	OC8E 03F6	=2656	ADD A,#-10 ;A>9 IS AN A OR 8 SEGMENT
	OC90 F67D	=2657	JC TBNONE ;JUMP IS A OR 8 SEGMENT
55		=2658	;
	OC92 7489	=2659	CALL SUM12C

ISIS-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
 QW4095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 36

```

5      LOC  OBJ      LINE      SOURCE STATEMENT

      =2661 ;
      OC95 8827      =2662      MOV      R0,#L6S1+3
      OC97 F0        =2663      MOV      A,BR0      ;GET PARITY DECODE CHARACTER
      OC98 68        =2664      ADD      A,R3      ;ADD 13TH CHARACTER
10     OC99 57        =2665      DA      A
      OC9A 530F      =2666      ANL      A,B0FH
      OC9C 9675      =2667      JNZ      TBERR6      ;JUMP IF MOD-10 IS BAD
      OC9E 83        =2668      RET
      =2669 ;
      =2670 ; ROUTINE: TBLK2
      =2671 ; FUNCTION: TRY FOR A VALID VERSION-0 BLOCK-2.
15     =2672 ; CHECK THAT L6 AND R6 HAVE ENOUGH DATA.
      =2673 ; CHECK THAT L6 IS A D-TAG.
      =2674 ; IF OK SO FAR, CALCULATE MOD-10 CHECK CHARACTER.
      =2675 ; IF STILL OK, RETURN WITH A=0.
      =2676 ; IF MOD-10 ERROR, CLEAR 486-CHAR SEGMENT COUNTERS AND
      =2677 ; CLEAR VERSION POINTER/FLAG.
      =2678 ; ENTRY: SCAN 1 BUFFER IS MAJORITY SCAN.
      =2679 ; R1 SETUP WITH MINUS THE MINIMUM NUMBER OF SCANS REQUIRED
20     =2680 ; EXIT: USER R0,R2,R3,A
      =2681 ; A=0 IF GOOD BLOCK
      =2682 ; A<0 IF NO BLOCK
      =2683 ;
      =2684 TBLK2: CALL  CK6TOT
      OCA1 C67D      =2685      JZ      TBNONE      ;JUMP IF NO LEFT OR NO RIGHT SEGMENTS
      =2686 ;
      =2687      MOV      R0,#L6S1+3
      OCA3 8827      =2688      MOV      A,BR0      ;GET PARITY DECODE BYTE
      OCA5 F0        =2689      ANL      A,#DECD
      OCA6 5320      =2690      JZ      TBNONE      ;JUMP IF NOT A D-TAG.
      OCA8 C67D      =2691 ;
      =2692      CALL  SUN12C
      OCAA 7489      =2693      JNZ      TBERR6      ;JUMP IF MOD-10 TEST FAILED
      OCAC 9675      =2694      RET
      OCAE 83        =2695 ;
      =2696 ; ROUTINE: TBLKE
      =2697 ; FUNCTION: TRY FOR A VALID UPC-E BLOCK.
30     =2698 ; CHECK THAT L6 HAS ENOUGH DATA
      =2699 ; CHECK THAT L6 IS AN E-TAG.
      =2700 ; CHECK THAT R6 HAS NO DATA.
      =2701 ; IF OK, RETURN WITH A=0.
      =2702 ; ELSE, CLEAR 486-CHAR SEGMENT COUNTERS AND
      =2703 ; CLEAR THE VERSION POINTER/FLAG.
      =2704 ; ENTRY: SCAN 1 BUFFER IS THE MAJORITY SCAN.
35     =2705 ; R1 SETUP WITH MINUS THE MINIMUM NUMBER OF SCANS REQUIRED
      =2706 ; EXIT: USES R0,A
      =2707 ; A=0 IF GOOD BLOCK.
      =2708 ; A<0 IF NO BLOCK.
      =2709 ;
      =2710 TBLKE: MOV      R0,#L6STOT
      OCAF 882D      =2711      MOV      A,BR0
      OC81 F0        =2712      ADD      A,R1
      OC82 69        =2713      JNC      TBNONE      ;JUMP IF < -R1 LEFT SEGMENTS
      OC83 E67D      =2714 ;
      =2715      MOV      R0,#L6S1+3
      OC85 8827      =2716      MOV      A,BR0      ;GET PARITY DECODE BYTE
      OC87 F0        =2717      ANL      A,#DECE
      OC88 5310      =2718      JZ      TBNONE      ;JUMP IF NOT AN E-SEG.
      OC8A C67D      =2719 ;
45     =2720      MOV      R0,#L6STOT
      OC8C 8835      =2721      MOV      A,BR0
      OC8E F0        =2722      JNZ      TBERR6      ;JUMP IF R6 SEGMENTS ARE PRESENT
      OC8F 9675      =2723 ;
      OCC1 83        =2724      RET
      =2725 ;
      =2726 ; ROUTINE: TBLCB
50     =2727 ; FUNCTION: TRY FOR A VALID EAN-8 BLOCK.
      =2728 ; IF ANY 6-CHAR SEGMENTS ARE PRESENT, CLEAR 4-CHAR
      =2729 ; SEGMENT COUNTERS AND VERSION POINTER/FLAG.
      =2730 ; CHECK THAT L4 AND R4 HAVE ENOUGH DATA.
      =2731 ; IF OK SO FAR, CALCULATE THE MOD-10 CHECK CHARACTER.
      =2732 ; IF STILL OK, RETURN WITH A=0.
      =2733 ; IF MOD-10 ERROR, CLEAR 4-CHAR SEGMENT COUNTERS AND
      =2734 ; CLEAR VERSION POINTER/FLAG.
      =2735 ; ENTRY: SCAN 1 BUFFER IS THE MAJORITY SCAN.
55     =2736 ; R1 SETUP WITH MINUS THE MINIMUM NUMBER OF SCANS REQUIRED

```

5

ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V4.2
CH095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 37

LOC	OBJ	LINE	SOURCE STATEMENT
		=2738 ;	A=0 IF GOOD BLOCK
		=2739 ;	A<>0 IF NO BLOCK
		=2740 ;	
10	OCC2 B820	=2741 TBLK6:	MOV R0,PL6STOT
	OCC4 F0	=2742	MOV A,BR0
	OCC5 9677	=2743	JNZ TBERR4 ;JUMP IF ANY 6L SEGMENTS
		=2744 ;	
	OCC7 B835	=2745	MOV R0,PR6STOT
	OCC9 F0	=2746	MOV A,BR0
	OCCA 9677	=2747	JNZ TBERR4 ;JUMP IF ANY 6R SEGMENTS
		=2748 ;	
15	OCCC B838	=2749	MOV R0,PL6STOT
	OCCE F0	=2750	MOV A,BR0
	OCCE 69	=2751	ADD A,R1
	OCDO E67D	=2752	JNC TBNONE ;JUMP IF < -R1 LEFT HALF SEGMENTS
		=2753 ;	
	OC02 B841	=2754	MOV R0,PR6STOT
	OC04 F0	=2755	MOV A,BR0
	OC05 69	=2756	ADD A,R1
20	OC06 E67D	=2757	JNC TBNONE ;JUMP IF < -R1 RIGHT HALF SEGMENTS
		=2758 ;	
	OC08 B836	=2759	MOV R0,PL4S1
	OC0A 145C	=2760	CALL MOD104
	OC0C A8	=2761	MOV R3,A ;SAVE LEFT SUM
	OC0D B83C	=2762	MOV R0,PR4S1
	OC0F 145C	=2763	CALL MOD104
	OCE1 68	=2764	ADD A,R3 ;RIGHT SUM + LEFT SUM
25	OCE2 57	=2765	DA A
	OCE3 530F	=2766	ANL A,#0FH
	OCE5 9677	=2767	JNZ TBERR4 ;JUMP IF MOD-10 IS BAD
	OCE7 83	=2768	RET
		=2769	*****
	OCE8 A3	=2770 TROPCC:	MOV A,DA
	OCE9 83	=2771	RET
30	OD00	=2772	ORG OD00H
		=2773 ;	*****
		=2774 ;	ROUTINE: TBLK6
		=2775 ;	FUNCTION: TRY FOR A VALID VERSION-D BLOCK-6.
		=2776 ;	CHECK THAT N(3), N(5) AND B(R) HAVE ENOUGH DATA.
		=2777 ;	IF THEY DO, CALCULATE MOD-10 CHECK CHARACTER.
		=2778 ;	IF OK, RETURN WITH A=0.
		=2779 ;	ELSE, CLEAR 4-CHAR SEGMENT COUNTERS AND
		=2780 ;	CLEAR THE VERSION POINTER/FLAG.
35		=2781 ;	ENTRY: SCAN 1 IS THE MAJORITY SCAN.
		=2782 ;	R1 SETUP WITH MINUS THE MINIMUM NUMBER OF SCANS REQUIRED
		=2783 ;	EXIT: USES R0,R2,R3,A
		=2784 ;	A=0 IF GOOD BLOCK
		=2785 ;	A<>0 IF NO BLOCK
		=2786 ;	
40	OD00 B477	=2787 TBERR4:	JMP TBERR4
	OD02 B475	=2788 TBERR6:	JMP TBERR6
	OD04 B47D	=2789 TBNONE:	JMP TBNONE
		=2790 ;	
	OD06 B853	=2791 TBLK6:	MOV R0,PR3STOT
	OD08 F0	=2792	MOV A,BR0
	OD09 69	=2793	ADD A,R1
	OD0A E604	=2794	JNC TBNONE ;JUMP IF < -R1 R3 SEGMENTS
		=2795 ;	
45	OD0C B85F	=2796	MOV R0,PR5STOT
	OD0E F0	=2797	MOV A,BR0
	OD0F 69	=2798	ADD A,R1
	OD10 E604	=2799	JNC TBNONE ;JUMP IF < -R1 R5 SEGMENTS
		=2800 ;	
	OD12 B841	=2801	MOV R0,PR4STOT
	OD14 F0	=2802	MOV A,BR0
	OD15 69	=2803	ADD A,R1
50	OD16 E604	=2804	JNC TBNONE ;JUMP IF < -R1 R4 SEGMENTS
		=2805 ;	
	OD18 B84E	=2806	MOV R0,PR3S1
	OD1A 145C	=2807	CALL MOD104
	OD1C A8	=2808	MOV R3,A
		=2809 ;	
	OD10 B85A	=2810	MOV R0,PR5S1
	OD1F 145C	=2811	CALL MOD104
55	OD21 68	=2812	ADD A,R3
	OD22 57	=2813	DA A

ISIS-II MCS-46/UP1-41 MACRO ASSEMBLER, V4.2
 GMA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 38

LOC	OBJ	LINE	SOURCE STATEMENT
		=2815 ;	
0024	B83C	=2816	MOV R0, #R4S1
0026	145C	=2817	CALL MOD104
0028	68	=2818	ADD A, R3
0029	57	=2819	DA A
002A	530F	=2820	ANL A, #0F0H
002C	9600	=2821	JNZ TBER6J
002E	83	=2822	RET
		=2823 ;	*****
		=2824 ;	ROUTINE: TBLK1
		=2825 ;	FUNCTION: TRY FOR A VALID VERSION-D BLOCK-1.
		=2826 ;	CHECK THAT L6, N(6) AND B(L) HAVE ENOUGH DATA.
		=2827 ;	CHECK THAT L6 IS A D-TAG
		=2828 ;	REVERSE 8L DATA
		=2829 ;	CALCULATE MOD-10 CHECK CHARACTER.
		=2830 ;	IF OK, RETURN WITH A=0.
		=2831 ;	ELSE, CLEAR 446-CHAR SEGMENT COUNTERS AND
		=2832 ;	CLEAR THE VERSION POINTER/FLAG.
		=2833 ;	ENTRY: SCAN 1 IS THE MAJORITY SCAN.
		=2834 ;	R1 SETUP WITH NIMUS THE MINIMUM NUMBER OF SCANS REQUIRED
		=2835 ;	EXIT: USES R0, R2, R3, A
		=2836 ;	A=0 IF GOOD BLOCK
		=2837 ;	A=0 IF NO BLOCK
		=2838 ;	
002F	B82D	=2839 TBLK1:	MOV R0, #L6STOT
0031	F0	=2840	MOV A, #R0
0032	69	=2841	ADD A, R1
0033	E604	=2842	JNC TBNONJ ;JUMP IF < -R L6 SEGMENTS
		=2843 ;	
0035	B827	=2844	MOV R0, #L6S1+3
0037	F0	=2845	MOV A, #R0 ;GET PARITY DECODE BYTE
0038	5320	=2846	ANL A, #FEDECD
003A	E604	=2847	JZ TBNONJ ;JUMP IF NOT A D-TAG
		=2848 ;	
003C	B865	=2849	MOV R0, #N6STOT
003E	F0	=2850	MOV A, #R0
003F	69	=2851	ADD A, R1
0040	E604	=2852	JNC TBNONJ ;JUMP IF < -R1 N6 SEGMENTS
		=2853 ;	
0042	B838	=2854	MOV R0, #L4STOT
0044	F0	=2855	MOV A, #R0
0045	69	=2856	ADD A, R1
0046	E604	=2857	JNC TBNONJ ;JUMP IF < -R1 8L SEGMENTS
		=2858 ;	
0048	B824	=2859	MOV R0, #L6S1
004A	1461	=2860	CALL MOD106
004C	A8	=2861	MOV R3, A
		=2862 ;	
004D	B860	=2863	MOV R0, #N6S1
004F	145C	=2864	CALL MOD104
0051	68	=2865	ADD A, R3
0052	57	=2866	DA A
0053	A8	=2867	MOV R3, A
		=2868 ;	
0054	B836	=2869	MOV R0, #L4S1 ;REVERSE 8L DATA
0056	F0	=2870	MOV A, #R0 ;GET CHAR 182
0057	47	=2871	SWAP A
0058	18	=2872	INC R0
0059	20	=2873	XCH A, #R0 ;GET CHAR 384, SAVE CHAR 221
005A	47	=2874	SWAP A
005B	C8	=2875	DEC R0
005C	A0	=2876	MOV #R0, A ;SAVE CHAR 483
		=2877 ;	
005D	145C	=2878	CALL MOD104
005F	68	=2879	ADD A, R3
0060	57	=2880	DA A
0061	530F	=2881	ANL A, #0F0H
0063	9602	=2882	JNZ TBER6J
0065	83	=2883	RET
		=2884 ;	*****
		=2885 ;	ROUTINE: TBLK3
		=2886 ;	FUNCTION: TRY FOR A VALID VERSION-D BLOCK-3.
		=2887 ;	CHECK THAT N(2) AND B(R) HAVE ENOUGH DATA.
		=2888 ;	IF THEY DO, CALCULATE MOD-10 CHECK CHARACTER.
		=2889 ;	IF OK, RETURN WITH A=0.
		=2890 ;	ELSE, CLEAR 4-CHAR SEGMENT COUNTERS AND

ISIS-II MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
 GM4095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 39

LOC	OBJ	LINE	SOURCE STATEMENT
5		=2892 ;	ENTRY: SCAN 1 IS THE MAJORITY SCAN.
		=2893 ;	R1 SETUP WITH MINUS THE MINIMUM NUMBER OF SCANS REQUIRED
		=2894 ;	EXIT: USES R0,R2,R3,A
		=2895 ;	A=0 IF GOOD BLOCK
		=2896 ;	A=0 IF NO BLOCK
		=2897 ;	
	0066 B840	=2898	TBLK3: MOV R0,#N2STOT
	0068 F0	=2899	MOV A,BR0
10	0069 69	=2900	ADD A,R1
	006A E604	=2901	JNC TBNONJ ;JUMP IF < -R1 N2 SEGMENTS
		=2902 ;	
	006C B841	=2903	MOV R0,#N4STOT
	006E F0	=2904	MOV A,BR0
	006F 69	=2905	ADD A,R1
	0070 E604	=2906	JNC TBNONJ ;JUMP IF < -R1 BR SEGMENTS
		=2907 ;	
15	0072 B848	=2908	MOV R0,#N2S1
	0074 145C	=2909	CALL MOD104
	0076 AB	=2910	MOV R3,A
		=2911 ;	
	0077 B83C	=2912	MOV R0,#N4S1
	0079 145C	=2913	CALL MOD104
	007B 68	=2914	ADD A,R3
	007C 57	=2915	DA A
20	007D 530F	=2916	ANL A,#0FH
	007F 9600	=2917	JNZ TBER4J
	0081 83	=2918	RET
		=2919 ;	*****
		=2920 ;	ROUTINE: TBLK4
		=2921 ;	FUNCTION: TRY FOR A VALID VERSION-0 BLOCK-4.
		=2922 ;	CHECK THAT N(5) AND N(1) HAVE ENOUGH DATA.
25		=2923 ;	IF THEY DO, CALCULATE MOD-10 CHECK CHARACTER.
		=2924 ;	IF OK, RETURN WITH A=0.
		=2925 ;	ELSE, CLEAR 4-CHAR SEGMENT COUNTERS AND
		=2926 ;	CLEAR THE VERSION POINTER/FLAG.
		=2927 ;	ENTRY: SCAN 1 IS THE MAJORITY SCAN.
		=2928 ;	R1 SETUP WITH MINUS THE MINIMUM NUMBER OF SCANS REQUIRED
		=2929 ;	EXIT: USES R0,R2,R3,A
		=2930 ;	A=0 IF GOOD BLOCK
		=2931 ;	A=0 IF NO BLOCK
30		=2932 ;	
	0082 B85F	=2933	TBLK4: MOV R0,#N5STOT
	0084 F0	=2934	MOV A,BR0
	0085 69	=2935	ADD A,R1
	0086 E604	=2936	JNC TBNONJ ;JUMP IF < -R1 N5 SEGMENTS
		=2937 ;	
	0088 B847	=2938	MOV R0,#N1STOT
35	008A F0	=2939	MOV A,BR0
	008B 69	=2940	ADD A,R1
	008C E604	=2941	JNC TBNONJ ;JUMP IF < -R1 N1 SEGMENTS
		=2942 ;	
	008E B85A	=2943	MOV R0,#N5S1
	0090 145C	=2944	CALL MOD104
	0092 AB	=2945	MOV R3,A
		=2946 ;	
40	0093 B842	=2947	MOV R0,#N1S1
	0095 145C	=2948	CALL MOD104
	0097 68	=2949	ADD A,R3
	0098 57	=2950	DA A
	0099 530F	=2951	ANL A,#0FH
	009B 9600	=2952	JNZ TBER4J
	009D 83	=2953	RET
		=2954 ;	*****
45		=2955 ;	ROUTINE: TBLK5
		=2956 ;	FUNCTION: TRY FOR A VALID VERSION-0 BLOCK-5.
		=2957 ;	CHECK THAT N(4) AND B(R) HAVE ENOUGH DATA.
		=2958 ;	IF THEY DO, CALCULATE MOD-10 CHECK CHARACTER.
		=2959 ;	IF OK, RETURN WITH A=0.
		=2960 ;	ELSE, CLEAR 4-CHAR SEGMENT COUNTERS AND
		=2961 ;	CLEAR THE VERSION POINTER/FLAG.
50		=2962 ;	ENTRY: SCAN 1 IS THE MAJORITY SCAN.
		=2963 ;	R1 SETUP WITH MINUS THE MINIMUM NUMBER OF SCANS REQUIRED
		=2964 ;	EXIT: USES R0,R2,R3,A
		=2965 ;	A=0 IF GOOD BLOCK
		=2966 ;	A=0 IF NO BLOCK
		=2967 ;	

1515-11 MCS-68/UP1-41 MACRO ASSEMBLER, V4.2
CH0095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 40

LOC	OBJ	LINE	SOURCE STATEMENT
5	00A0 F0	=2969	MOV A,3R0
	00A1 69	=2970	ADD A,R1
	00A2 E604	=2971	JNC TBNONJ ;JUMP IF < -R1 N4 SEGMENTS
		=2972 ;	
	00A4 8841	=2973	MOV R0,#R4STOT
	00A6 F0	=2974	MOV A,3R0
	00A7 69	=2975	ADD A,R1
	00A8 E604	=2976	JNC TBNONJ ;JUMP IF < -R1 8R SEGMENTS
		=2977 ;	
10	00AA 8854	=2978	MOV R0,#R4S1
	00AC 145C	=2979	CALL MOD104
	00AE AB	=2980	MOV R3,A
		=2981 ;	
	00AF 883C	=2982	MOV R0,#R4S1
	00B1 145C	=2983	CALL MOD104
	00B3 68	=2984	ADD A,R3
	00B4 57	=2985	DA A
15	00B5 530F	=2986	ANL A,#0FH
	00B7 9600	=2987	JNZ TBER4J
	00B9 83	=2988	RET
		=2989 ;	
		=2990 ;	ROUTINE: TBLK7
		=2991 ;	FUNCTION: TRY FOR A VALID VERSION-D BLOCK-7.
		=2992 ;	CHECK THAT N(3), N(6) AND N(1) HAVE ENOUGH DATA.
20		=2993 ;	IF THEY DO, CALCULATE MOD-10 CHECK CHARACTER.
		=2994 ;	IF OK, RETURN WITH A=0.
		=2995 ;	ELSE, CLEAR 4-CHAR SEGMENT COUNTERS AND
		=2996 ;	CLEAR THE VERSION POINTER/FLAG.
		=2997 ;	ENTRY: SCAN 1 IS THE MAJORITY SCAN.
		=2998 ;	R1 SETUP WITH MINUS THE MINIMUM NUMBER OF SCANS REQUIRED
		=2999 ;	EXIT: USES R0,R2,R3,A
		=3000 ;	A=0 IF GOOD BLOCK
25		=3001 ;	A=0 IF NO BLOCK
		=3002 ;	
	00BA 8853	=3003	TBLK7: MOV R0,#R3STOT
	00BC F0	=3004	MOV A,3R0
	00BD 69	=3005	ADD A,R1
	00BE E604	=3006	JNC TBNONJ ;JUMP IF < -R1 R3 SEGMENTS
		=3007 ;	
	00C0 8865	=3008	MOV R0,#R6STOT
30	00C2 F0	=3009	MOV A,3R0
	00C3 69	=3010	ADD A,R1
	00C4 E604	=3011	JNC TBNONJ ;JUMP IF < -R1 R6 SEGMENTS
		=3012 ;	
	00C6 8847	=3013	MOV R0,#R1STOT
	00C8 F0	=3014	MOV A,3R0
	00C9 69	=3015	ADD A,R1
35	00CA E604	=3016	JNC TBNONJ ;JUMP IF < -R1 R1 SEGMENTS
		=3017 ;	
	00CC 884E	=3018	MOV R0,#R3S1
	00CE 145C	=3019	CALL MOD104
	00D0 AB	=3020	MOV R3,A
		=3021 ;	
	00D1 8860	=3022	MOV R0,#R6S1
	00D3 145C	=3023	CALL MOD104
40	00D5 68	=3024	ADD A,R3
	00D6 57	=3025	DA A
	00D7 AB	=3026	MOV R3,A
		=3027 ;	
	00D8 8842	=3028	MOV R0,#R1S1
	00DA 145C	=3029	CALL MOD104
	00DC 68	=3030	ADD A,R3
	00DD 57	=3031	DA A
45	00DE 530F	=3032	ANL A,#0FH
	00E0 9600	=3033	JNZ TBER4J
	00E2 83	=3034	RET
		=3035 ;	
	00E3 A3	=3036	TROPED: MOV A,2A
	00E4 83	=3037	RET
	00E0	=3038	ORG 00E0H
		=3039 ;	INCLUDE(:F1:SLRDTG.SRC)
50		=3040 ;	
		=3041 ;	FILE: SLRDTG.SRC 10-08-86 15:55 BOB ACTIS
		=3042 ;	ROUTINE: ROTAG FOR THE 750SL **THIS IS THE MAIN PROGRAM**
		=3043 ;	
	00E0 FC	=3044	ROTAG: MOV A,R4 ;CLEAR THE SCAN FLAGS EXCEPT FOR

1515-II MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
 QAD95 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 41

LOC	OBJ	LINE	SOURCE STATEMENT
5	0E03 AC	=3046	MOV R4,A
	0E04 B208	=3047	JBS RDT05 ;BUFMAN REQUEST
		=3048 ;	;JUMP IF THE BUFMAN REQUEST FLAG IS SET
10	0E06 9475	=3049	CALL CLRVER
	0E08 140C	=3050 RDT05: CALL CLRSHB	;CLEAR THE VERSION POINTER/FLAG & DATA
		=3051 ;	;CLEAR THE SCAN BUFFER
		=3052 ;	WAIT FOR THE LABEL TO GO AWAY. CHECK MOTOR SPEED.
		=3053 ;	
	0E0A D484	=3054 RDT10: CALL NTRCHK	;CHECK MOTOR SPEED
	0E0C 3400	=3055	CALL NCOMH
	0E0E B619	=3056	JNI RDT20
		=3057 ;	;SERVICE THE I/F AND BUFMAN
15	0E10 FF	=3058	MOV A,R7
	0E11 960A	=3059	JNZ RDT10
		=3060 ;	;GET THE "SEG SEEN" TIMER
		=3061	MOV R5,#EDRDLY
		=3062 ;	;SET THE DOUBLE READ TIMER
	0E13 B016	=3063 RDT15: MOV R7,#EGDLTV	;SET THE GO-LT ON TIMER
	0E15 BF2A	=3064	JMP RDT30
	0E17 C41F	=3065 ;	;GO WAIT FOR A LABEL TO READ
20		=3066 ;	RESET THE SYNCAP OR SERVICE SDATA
		=3067 ;	
	0E19 1444	=3068 RDT20: CALL CKFCA	;RESET THE SEG, ALSO CHECK FOR SDATA
	0E1B 3400	=3069	CALL NCOMH
	0E1D C40A	=3070	JMP RDT10
		=3071 ;	;SERVICE THE I/F AND BUFMAN. ALSO
		=3072 ;	;DELAY FOR FCA TO RESET
		=3073 ;	WAIT FOR A LABEL TO READ. CHECK GO-LT ON TIME AND MOTOR SPEED.
25	0E1F FC	=3074 RDT30: MOV A,R4	
	0E20 4301	=3075	ORL A,#ESONG
	0E22 AC	=3076	MOV R4,A
		=3077 ;	;SET THE SCAN FLAG SO CKFCA WILL
		=3078 RDT35: CALL NCOMH	;PUT THE SEGMENT INTO THE SCAN BUFFER
	0E23 3400	=3079	CALL CKFCA
	0E25 1444	=3080	MOV R0,#SCANBUF+3
	0E27 B823	=3081	MOV A,R0
30	0E29 FD	=3082	JNZ RDT40
	0E2A 9637	=3083 ;	;GET THE SCAN BUF PARITY DECODE BYTE
		=3084	CALL NTRCHK
	0E2C D484	=3085 ;	;CHECK MOTOR SPEED
		=3086	MOV A,R7
	0E2E FF	=3087	JNZ RDT35
	0E2F 9623	=3088	ANL P1,#Z55-EGDLT
35	0E31 990F	=3089	ORL P1,#EBDLT
	0E33 8940	=3090	JMP RDT35
	0E35 C423	=3091 ;	;JUMP IF GO-LT TIMER = 0
		=3092 ;	;GO-LT OFF
		=3093 ;	;GO-LT ON
		=3094 RDT40: CALL NCOMH	;STILL WAITING FOR A LABEL
		=3095	ANL P1,#Z55-EGDLT
	0E37 3400	=3096	ORL P1,#EBDLT
	0E39 990F	=3097	MOV A,R4
40	0E3B 8940	=3098	JBS RDTAG
	0E3D FC	=3099	JMP RDT60
	0E3E B200	=3100 ;	;JUMP IF THE BUFMAN REQUEST FLAG IS SET
	0E40 C448	=3101 ;	;GO PROCESS THE FIRST SEGMENT
		=3102 ;	COLLECT AND PROCESS SEGMENTS. CHECK MOTOR SPEED.
		=3103 RDT50: CALL NTRCHK	;CHECK MOTOR SPEED
45	0E42 D484	=3104	CALL NCOMH
	0E44 3400	=3105	CALL CKFCA
	0E46 1444	=3106 RDT60: CALL PROCSC	;GET SEGMENTS IF ANY
	0E48 5404	=3107	SEL MB0
	0E4A E5	=3108	CALL GCNTS
	0E4B D400	=3109	SEL MB1
	0E4D F5	=3110	JZ RDT70
	0E4E C653	=3111 ;	;JUMP IF ENOUGH SEGMENTS
50	0E50 FF	=3112	MOV A,R7
	0E51 9642	=3113	JNZ RDT50
		=3114 ;	;JUMP IF SEGMENTS RECENTLY SEEN
		=3115 ;	TRY FOR A VALID LABEL
		=3116 ;	
	0E53 3400	=3117 RDT70: CALL NCOMH	
	0E55 9400	=3118	CALL VERTAG
	0E57 C660	=3119	JZ GOODRD
55		=3120 ;	;JUMP IF A GOOD VERSION WAS FOUND
	0E59 FF	=3121	MOV A,R7

ISIS-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
GMA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 42

```

5      LOC OBJ      LINE      SOURCE STATEMENT

      OESC 3400      =3123 ;
      OESE C400      =3124 BADRD: CALL NCOMH
                      =3125 JMP RDTAG ;GO START OVER. NO BUFMAN REQUEST.
                      =3126 ;
      OE60 3400      =3127 GOODRD: CALL NCOMH
      OE62 FD        =3128 MOV A,R5 ;GET THE DOUBLE READ TIMER
      OE63 C660      =3129 JZ GOOD10 ;JUMP IF DR TIMER=0 (OK TO DOUBLE READ)
                      =3130 ;
      OE65 E5        =3131 SEL N80
      OE66 F408      =3132 CALL DRSUMT ;CALCULATE AND TEST THE DOUBLE READ SUM
      OE68 F5        =3133 SEL N81
      OE69 9671      =3134 JNZ GOOD20 ;JUMP IF OLD<NEW (NO DOUBLE READ)
      OE6A C400      =3135 JMP RDTAG ;JUMP IF DOUBLE READ TOO SOON
                      =3136 ;
      OE6D E5        =3137 GOOD10: SEL N80
      OE6E F408      =3138 CALL DRSUMT ;CALCULATE AND SAVE THE LABEL SUM
      OE70 F5        =3139 SEL N81
      OE71 998F      =3140 GOOD20: ANL P1,#255-EBDLT ;BD-LT OFF
      OE73 8920      =3141 ORL P1,#EBDLT ;CD-LT ON
      OE75 D5        =3142 SEL R81
      OE76 FB        =3143 MOV A,R3 ;TONE ENABLE FLAG
      OE77 9678      =3144 JNZ GOOD30 ;JUMP IF TONE DISABLED
      OE79 BE04      =3145 MOV R6,#EGDTON ;GOOD TONE LENGTH
      OE7A C5        =3146 GOOD30: SEL R80
                      =3147 ;
      OE7C FC        =3148 RDT90: MOV A,R6
      OE7D 4320      =3149 ORL A,#EBFREQ ;SET THE BUFMAN REQUEST FLAG
      OE7F AC        =3150 MOV R4,A
      OE80 F409      =3151 CALL BUFMAN
      OE82 C400      =3152 JMP RDTAG
                      =3153 ;
      OE84 0A        =3154 ; CHECK MOTOR UP2SPD* SIGNAL (UP TO SPEED & OVER SPEED)
      OE85 37        =3155 NTRCHK: IN A,P2
      OE86 8289      =3156 CPL A
      OE88 83        =3157 JBS NTRC10 ;JUMP IF MOTOR SPEED PROBLEM
                      =3158 RET ;RETURN IF OK
                      =3159 ;
      OE89 8F64      =3160 NTRC10: MOV R7,#100 ;SET TIMER FOR 2 SECONDS
      OE8B FF        =3161 NTRC20: MOV A,R7
      OE8C 9688      =3162 JNZ NTRC20 ;WAIT IN CASE OF SPURIOUS ERROR
                      =3163 ;
      OE8E 0A        =3164 IN A,P2
      OE8F 37        =3165 CPL A
      OE90 8293      =3166 JBS NTRERR ;JUMP IF STILL A PROBLEM AFTER WAIT
      OE92 83        =3167 RET ;RETURN IF OK
                      =3168 ;
      OE93 8910      =3169 ; COME HERE IF THERE IS A MOTOR PROBLEM DURING RDTAG
      OE95 99D0      =3170 NTRERR: ORL P1,#ELASDB ;LASER OFF
      OE97 8804      =3171 ANL P1,#255-(ENTREB+EBDLT) ;MOTOR OFF, GREEN LIGHT OFF
      OE99 E5        =3172 MOV R0,R4
      OE9A 747F      =3173 SEL N80
      OE9C F5        =3174 CALL TERRMT ;GIVE 4 BEEPS FOR A MOTOR ERROR
                      =3175 SEL N81
                      =3176 ;
      OE9D 8F05      =3177 ; STICK HERE WITH THE RED LIGHT FLASHING
      OE9F FF        =3178 NTRC20: MOV R7,#5 ;SET TIMER FOR 100MS
      OEA0 969F      =3179 NTRC30: MOV A,R7
                      =3180 JNZ NTRC30 ;WAIT BETWEEN LIGHT TOGGLES
                      =3181 ;
      OEA2 09        =3182 IN A,P1
      OEA3 998F      =3183 ANL P1,#255-EBDLT ;RED LIGHT OFF
      OEA5 0290      =3184 JBS NTRC20 ;JUMP IF THE RED LIGHT WAS ON
      OEA7 8940      =3185 ORL P1,#EBDLT ;RED LIGHT ON
      OEA9 C490      =3186 JMP NTRC20
                      =3187 ;*****
      OEAB A3        =3188 TROPGE: MOV A,8A
      OEAC 83        =3189 RET
      OF00           =3190 ORG -- OF00H
      OF00           =3191 $ INCLUDE(:"F1:BUFMAN.SRC")
                      =3192 ;*****
      OF00           =3193 ; FILE: BUFMAN.SRC 10-08-86 16:15 BOB ACTIS
                      =3194 ;*****
      OF00           =3195 ; ROUTINE: BUF12C
      OF00           =3196 ; FUNCTION: LOAD 12 CHARACTERS INTO THE COMMUNICATIONS BUFFER.
      OF00           =3197 ; (LOAD L6 AND R6 DATA)
      OF00           =3198 ; ENTRY: R1 = NEXT AVAILABLE COMM BUFFER BYTE.

```

ISIS-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
CHAO95 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 43

```

5      LOC OBJ      LINE      SOURCE STATEMENT

      =3200 ;          L6 AND R6 DATA MOVED TO COMM BUFFER.
      =3201 ;
10     0F00 B824     =3202 BUF12C: MOV      R0,#L6S1
      0F02 142A     =3203          CALL    MOV38Y
      0F04 B82E     =3204          MOV      R0,#R6S1
      0F06 142A     =3205          CALL    MOV38Y
      0F08 83       =3206          RET
      =3207 ;
      =3208 ;          ROUTINE:  BUFMAN
      =3209 ;          FUNCTION:  IF BUFMAN REQUEST FLAG IS NOT SET,
15     =3210 ;          THEN RETURN,
      =3211 ;          ELSE IF COMM BUF IS BUSY
      =3212 ;          THEN IF NO VALID VERSIONS, CLR REQ FLAG. RETURN
      =3213 ;          ELSE PROCESS MESSAGE BUFFER REQUEST.
      =3214 ;          ENTRY:  R80
      =3215 ;          R6 VERSION FLAGS SETUP
      =3216 ;          EXIT:  USES R0,R1,A
      =3217 ;
      =3218 ;          SPECIAL CHARACTERS:
20     =3219 ;          DATA "C" = FILLER CHARACTER
      =3220 ;          BYTE OFXN = LABEL TYPE TERMINATOR FLAG
      =3221 ;          BYTE OCCN = TERMINATION (ETRM8Y)
      =3222 ;
      0F09 FC       =3223 BUFMAN: MOV      A,R6
      0F0A B200     =3224          JBS     BUFN10          ;JUMP IF BUFMAN REQUEST FLAG IS SET
      0F0C 83       =3225          RET
      0F0D 72A3     =3226 BUFN10: JBS     BUFN93          ;JUMP IF COMM BUFFER IS BUSY
25     =3227 ;
      0F0F B967     =3228 BUFN20: MOV      R1,#SBUF        ;SEND BUFFER START ADDRESS
      0F11 FE       =3229          MOV      A,R6          ;GET VERSION FLAGS
      0F12 530F     =3230          ANL     A,#OFXN        ;MASK VERSION POINTER
      0F14 0300     =3231          ADD     A,R0          ;SETUP CARRY FOR DA
      0F16 57       =3232          DA      A
      0F17 92A8     =3233          JBS     BUFN94          ;JUMP IF POINTER > 9.  ILLEGAL VERSION.
30     =3234 ;
      =3235 ;          ADD     A,#LOW BUFTBL
      0F19 031C     =3236          JNPP    0A
      0F1B 83       =3237 BUFTBL: DB      LOW BUFN94      ;ILLEGAL VERSION. THIS WAS MISSEN
      0F1C A8       =3238          DB      LOW BUFMA
      0F1D 26       =3239          DB      LOW BUFMA
      0F1E 2A       =3240          DB      LOW BUFMA
      0F1F 33       =3241          DB      LOW BUFMA
      0F20 43       =3242          DB      LOW BUFMA
35     0F21 47       =3243          DB      LOW BUFMA
      0F22 55       =3244          DB      LOW BUFMA
      0F23 58       =3245          DB      LOW BUFMA
      0F24 65       =3246          DB      LOW BUFMA
      0F25 79       =3247 ;
      0F26          =3248 BUFMA  EQU      $
      0F26 F400     =3249 BUFN12: CALL    BUF12C
40     0F28 E48F     =3250          JMP     BUFN90
      =3251 ;
      0F2A 23C0     =3252 BUFN13: MOV      A,#OCCN        ;FILLER CHARACTER
      0F2C B827     =3253          MOV      R0,#L6S1+3    ;PARITY DECODE CHARACTER ADDRESS
      0F2E 30       =3254          XCHD    A,DRO        ;PUT PARITY DECODE CHAR INTO A WITH FILLER
      0F2F A1       =3255          MOV      DR1,A        ;PUT FILLER & CHAR INTO SEND BUFFER
      0F30 19       =3256          INC     R1            ;NEXT AVAILABLE BUFFER LOCATION
      0F31 E426     =3257          JMP     BUFN12        ;GO DO THE NEXT 12 CHARS
45     =3258 ;
      0F33 81C0     =3259 BUFME:  MOV      DR1,#OCCN      ;FILLER W/ E-R/S-0 DIGIT
      0F35 19       =3260          INC     R1
      0F36 B824     =3261          MOV      R0,#L6S1
      0F38 142A     =3262          CALL    MOV38Y
      0F3A 23C0     =3263          MOV      A,#OCCN
      0F3C B827     =3264          MOV      R0,#L6S1+3    ;FILLER CHARACTER
      0F3E 30       =3265          XCHD    A,DRO        ;PARITY DECODE CHARACTER ADDRESS
      0F3F A1       =3266          MOV      DR1,A        ;PUT PARITY DECODE CHAR INTO A WITH FILLER
50     0F40 19       =3267          INC     R1            ;PUT FILLER & CHAR INTO SEND BUFFER
      0F41 E48F     =3268          JMP     BUFN90
      =3269 ;
      0F43          =3270 BUFMS  EQU      $
      0F43 B836     =3271 BUFMSL: MOV      R0,#L6S1
      0F45 E471     =3272          JMP     BUFMSR
      =3273 ;
55     0F47 B824     =3274 BUFND1: MOV      R0,#L6S1
      0F49 142A     =3275          CALL    MOV38Y

```

ISIS-11 MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
QWAD95 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 44

5

10

15

20

25

30

35

40

45

50

55

LOC	OBJ	LINE	SOURCE STATEMENT
0F40	1426	=3277	CALL MOV2BY
0F4F	B836	=3278	MOV R0,#1451 ;ALREADY SWAPPED
0F51	1426	=3279	CALL MOV2BY
0F53	E48F	=3280	JMP BUFH90
		=3281 ;	
0F55	F400	=3282	BUFH02: CALL BUF12C
0F57	B848	=3283	MOV R0,#1251
0F59	E471	=3284	JMP BUFH0R
		=3285 ;	
0F5B	F400	=3286	BUFH03: CALL BUF12C
0F5D	B84E	=3287	MOV R0,#1351
0F5F	1426	=3288	CALL MOV2BY
0F61	B85A	=3289	MOV R0,#1551
0F63	E471	=3290	JMP BUFH0R
		=3291 ;	
0F65	F400	=3292	BUFH04: CALL BUF12C
0F67	B85A	=3293	MOV R0,#1551
0F69	1426	=3294	CALL MOV2BY
0F6B	B842	=3295	MOV R0,#1151
0F6D	1426	=3296	CALL MOV2BY
0F6F	B854	=3297	MOV R0,#1451
0F71	1426	=3298	BUFH0R: CALL MOV2BY
0F73	B83C	=3299	MOV R0,#1451
0F75	1426	=3300	CALL MOV2BY
0F77	E48F	=3301	JMP BUFH90
		=3302 ;	
0F79	F400	=3303	BUFH05: CALL BUF12C
0F7B	B854	=3304	MOV R0,#1451
0F7D	1426	=3305	CALL MOV2BY
0F7F	B83C	=3306	MOV R0,#1451
0F81	1426	=3307	CALL MOV2BY
0F83	B84E	=3308	MOV R0,#1351
0F85	1426	=3309	CALL MOV2BY
0F87	B860	=3310	MOV R0,#1451
0F89	1426	=3311	CALL MOV2BY
0F8B	B842	=3312	MOV R0,#1151
0F8D	1426	=3313	CALL MOV2BY
		=3314 ;	
0F8F	FE	=3315	BUFH90: MOV A,R6 ;GET VERSION FLAG
0F90	43F0	=3316	ORL A,#0F0H ;PUT IN TERMINATION FLAG NIBBLE
0F92	A1	=3317	MOV R1,A ;PUT TERMINATION FLAG BYTE IN BUFFER
0F93	19	=3318	INC R1
		=3319 ;	
0F94	81CC	=3320	MOV R1,#PTRMBY ;LOAD THE DATA TERMINATION CHARACTER
0F96	8966	=3321	MOV R1,#SBPNT ;SEND BUFFER POINTER ADDRESS
0F98	81CE	=3322	MOV R1,#SBSTRT ;PUT PACKED DATA START ADRS IN POINTER
0F9A	9475	=3323	CALL CLRVER ;CLEAR THE VERSION POINTER/FLAG & DATA
		=3324 ;	
0F9C	FC	=3325	MOV A,R4
0F9D	530F	=3326	ANL A,#255-EBFRE0 ;CLEAR THE BUFMAN REQUEST FLAG
0F9F	4308	=3327	ORL A,#ESBFUL ;SET THE SEND BUFFER FULL BIT
0FA1	AC	=3328	MOV R4,A
0FA2	83	=3329	RET
		=3330 ;	
0FA3	FE	=3331	BUFH93: MOV A,R6
0FA4	530F	=3332	ANL A,#0F0H
0FA6	96AC	=3333	JNZ BUFH95 ;JUMP IF A VALID VERSION
		=3334 ;	
0FAB	FC	=3335	BUFH94: MOV A,R4
0FA9	530F	=3336	ANL A,#255-EBFRE0 ;CLEAR THE BUFMAN REQUEST FLAG
0FAB	AC	=3337	MOV R4,A
		=3338 ;	
0FAC	83	=3339	BUFH95: RET
		3340 ;	
0FAD	A3	3341	TROPGE: MOV A,R4
0FAE	83	3342	RET
		3343 ;	
0FF7		3344	ORG OFF7H
		3345 ;	
		3346 ;	CHECKSUM BYTE
		3347 ;	
0FF7	25	3348	DB 25H
		3349 ;	
		3350 ;	DATE
		3351 ;	
0FFB	02	3352	DB 02H,20H,88H

PAGE 45

```

LOC OBJ          LINE          SOURCE STATEMENT
OFFA 88          3353 ;
                  3354 ; PART NUMBER
                  3355 ;
10 OFFB 52       3356      DB      "R,9CM,01N,53N
OFFC 96
OFFD 01
OFFE 53
                  3357 ;
                  3358 ; REVISION
                  3359 ;
OFFF 41          3360      DB      "A"
                  3361 -----
15                  3362      END

```

[illegible]

ISIS-II MCS-48/UP1-41 MACRO ASSEMBLER, V4.2
 QNA095 ASSEMBLED 2/22/88 BY BLAKE ISAACS

PAGE 46

TSC10 04A6	TSC11 04B8	TSC12 04B4	TSC20 047E	TSCBUF 0050	TSCNT 0462	TSEG1 0030	TSEG2 0034
TST31 0251	TST32 0255	TST41 0262	TST42 0266	TTA90 02A7	TTACKK 02A9	TTAG 020E	TTARET 0369
TTATA8 0200	UNPK1 0210	UNPK2 0222	VER13 0C60	VER8 0C5E	VERA 0C61	VERD1 0C58	VERD2 0C57
VERD3 0C56	VERD4 0C55	VERD5 0C54	VERE 0C5F	VERFLG 0006	VERT05 0C2C	VERT10 0C35	VERT15 0C3E
VERT20 0C48	VERT80 0C58	VERT90 0C62	VERTAG 0C00	VRICBUF 0079			

ASSEMBLY COMPLETE, NO ERRORS

10

ISIS-II ASSEMBLER SYMBOL CROSS REFERENCE, V2.1

PAGE 1

APL342 1625#	1639	1648	1711																
BADRD 3126#																			
BF4CMT 251#	1514																		
BF4CST 211#	251	1513																	
BF6CMT 209#	1502																		
BF6CST 199#	209	1501																	
BUF12C 3202#	3249	3282	3286	3292	3303														
BUF110 3224	3226#																		
BUF112 3249#	3257																		
BUF113 3259	3252#																		
BUF120 3228#																			
BUF18 3241	3270#																		
BUF18L 3271#																			
BUF18R 3272	3284	3290	3298#																
BUF190 3250	3268	3280	3301	3315#															
BUF193 3226	3331#																		
BUF194 3253	3237	3335#																	
BUF195 3333	3339#																		
BUF1A 3238	3248#																		
BUF1AH 1832	3151	3223#																	
BUF1D1 3242	3274#																		
BUF1D2 3243	3282#																		
BUF1D3 3244	3286#																		
BUF1D4 3245	3292#																		
BUF1D5 3246	3303#																		
BUF1E 3240	3259#																		
BUF1EL 3235	3237#																		
CG6T10 2442#	2448																		
CG6T20 2441	2445#																		
CG6T01 2438#	2607	2646	2684																
CCCR05 1251	1254#																		
CCCR10 1269#																			
CCCR20 1257	1275#																		
CCCR30 1262	1285#																		
CCCR40 1287	1292	1296#																	
CCCR45 1301#																			
CCCR50 1282	1308#																		
CCCR60 1321	1330#																		
CCCR70 1316	1337#																		
CCCR80 1345	1354#																		
CCCRMG 1267	1273	1278	1294	1299	1304	1306	1311	1326	1328	1333	1335	1340	1350	1352	1357	1358	1363#		
CCCRMK 1272	1283	1288	1293	1305	1327	1334	1351	1360#											
CCCRMS 1246#	3108																		
CCFC10 1734	1737#																		
CCFC20 1739	1746#																		
CCFC80 1777	1781#																		
CCFC90 1748	1779	1783#																	
CCFC95 1741	1747	1787#																	
CCFCA 416	1734#	3068	3079	3105															
CCMAJ 2306#	2466	2473	2482																
CCMAJ9 2312	2318#																		
CCM15H 2334#	2470	2477	2489																
CCMS30 2352	2369#																		
CCMS80 2358	2364	2371	2376#																
CCMS90 2357	2347	2379#																	
CCRCV 1824	1906#																		
CCRCV1 1909#																			
CCRCV2 1910	1914#																		
CCRCV3 1916	1920#																		
CCRCV4 1922	1928#																		
CCRCV5 1930	1937#																		
CCRCV6 1939	1943#																		
CCRCV9 1946#																			
CLRASC 1513#	2624																		
CLRASC 408	1501#	2623																	
CLRBAN 451#	652																		
CLRSBF 1544#	1898																		
CLRSH1 1528#	1530	1554																	
CLRSH8 1525#	1778	2125	3050																
CLRT00 1503	1515	1527#																	
CLRYER 2622#	3049	3323																	
CCMRST 155#	1938																		
CCMFIC 269#	881	2517	2536	2549															
CPARTY 935	1132																		
D1BEEP 154#	1929																		
D1SCAN 152#	1915																		
DREH13 1417	1427#																		
DREH1X 1457	1466#																		
DREH1X 1451	1461#																		

ISIS-II ASSEMBLER SYMBOL CROSS REFERENCE, V2.1

PAGE 2

[illegible]

ISIS-11 ASSEMBLER SYMBOL CROSS REFERENCE, V2.1

PAGE 3

[illegible]

ISIS-II ASSEMBLER SYMBOL CROSS REFERENCE, V2.1

PAGE 4

[illegible]

ISIS-11 ASSEMBLER SYMBOL CROSS REFERENCE, V2.1

PAGE 5

[illegible]

ISIS-11 ASSEMBLER SYMBOL CROSS REFERENCE, V2.1

PAGE 6

	TERRUT	777	786#	3174					
	TEST1	574#	605						
	TEST2	576#	604						
5	TEST3	579#	579						
	TEST4	592#	592						
	TNO00	909#	937	942	961	964	968	992	
	TNO05	378	915#	916	925	945	963		
	TNO10	915	921#						
	TNO12	933#							
	TNO14	932	935#						
	TNO15	939	949#						
10	TNO20	950	954#						
	TNO29	953	959#						
	TNO30	962	968#						
	TNO50	978#	988						
	TNO55	979	983#						
	TNO59	982	985#						
	TNSHD	974	987	991	1129#	1130			
	TIME02	309	312#						
15	TIME05	314	317#						
	TIME10	318	321#	348					
	TIME20	322#	351						
	TIME30	306	327#						
	TIME40	329	332#						
	TIME45	337	340#						
	TIME50	342	345#						
20	TIME60	333	346	350#					
	TIMER	303#							
	TIMREG	180#							
	TIKTRP	287#							
	TNOT10	674#	675						
	TNOT20	683#							
	TNOT22	684#	696	699					
	TNOT24	685#	688						
25	TNOT28	692	701#						
	TNOT40	705#	706						
	TNOT50	679	712#						
	TNOT60	713#	728	731					
	TNOT80	714#	717						
	TNOT90	709	724	734#					
	TNOT95	720	740#						
	TNOTOR	669#	770						
30	TNMA10	1228#	1229						
	TNMA1T	1032	1223#						
	TOMCNT	189#							
	TOKLTN	190#							
	TPON	747#	840						
	TPON15	857#	858						
	TPON20	748	752#						
	TPON30	753	757#						
35	TPON40	760	764#						
	TPON50	766	770#						
	TPON60	771	775#						
	TPON90	750	755	762	768	773	777#		
	TPORET	775	841#						
	TRAH	457#	752						
	TRAH10	458#	460						
	TRAH20	464#	467						
40	TRAH30	471#	479						
	TRAH40	483#	491						
	TRAH50	495#	503						
	TRAH60	507#	515						
	TRAH8	466	487	511	518#				
	TRARET	518	753#						
	TRO10	406#	420						
	TRO20	415#	418						
45	TRO50	415	426#						
	TROESH	393#	747						
	TROPG0	382#	433	--					
	TROPG1	434	521#						
	TROPG2	435	659#						
	TROPG3	436	887#						
	TROPG4	437	1109#						
50	TROPG5	438	1232#						
	TROPG6	439	1375#						
	TROPG7	440	1485#						
	TROPG8	441	1792#						
	TROPG9	442	2057#						
	TROPGA	443	2243#						

55

ISIS-II ASSEMBLER SYMBOL CROSS REFERENCE, V2.1

PAGE 7

	TROPCH	444	2496#				
	TROPCC	445	2770#				
	TROPCD	446	3036#				
	TROPCE	447	3188#				
5	TROPCE	448	3341#				
	TRORET	422	748#				
	TROTAB	407	433#				
	TAYBLK	2519	2522#				
	TSCDZ	1037#	1039				
	TSCD4	1048#	1052				
	TSCD5	1050#					
10	TSCD6	1051#	1060	1096	1107		
	TSCD7	1059#	1084				
	TSCD8	1050	1068#				
	TSCD9	1066	1077#				
	TSC10	1079	1086#				
	TSC11	1067#	1070				
	TSC12	1062#	1071				
	TSC20	1056#					
15	TSCBUF	169#	1064	1068	1082	1086	1199
	TSCMT	969	1030#				
	TSEG1	165#	374	911	1077	1089	
	TSEG2	166#	1080	1094			
	TST31	582	585#				
	TST32	584	589#				
	TST41	596	599#				
	TST42	598	603#				
20	TTA90	627	639	654#			
	TTACHK	635	656#				
	TTAG	534#	764				
	TTARET	654	765#				
	TTATAB	532#	537				
	UNPK1	542#	565				
	UNPK2	546#	563				
	VER13	2526	2585#				
25	VER8	2545	2583#				
	VERA	2523	2586#				
	VERD1	2542	2575#				
	VERD2	2561	2574#				
	VERD3	2558	2573#				
	VERD4	2568	2572#				
	VERD5	2565	2571#				
	VERE	2533	2584#				
30	VERFLG	179#					
	VERT05	2538	2541#				
	VERT10	2529	2548#				
	VERT15	2551	2554#				
	VERT20	2555	2564#				
	VERT80	2580#					
	VERT90	2588#					
35	VERTAG	2510#	3118				
	VRKBUF	263#	1661	1676	1685	1696	1705

CROSS REFERENCE COMPLETE

It can be seen therefore that the use of a common interface according to the present invention in a combined scanner and scale system provides significant advantages over prior art systems in which the scanner and the scale do not share an interface. Only a single connector cable and only a single port of the cash register system are required to connect both the scale and the scanner to the cash register system. Further the costs associated with the scanner and the scale having dedicated interface circuits are eliminated by the sharing of an interface circuit according to the present invention.

Having described the invention in detail and by reference to the preferred embodiment thereof, it will be apparent that other modifications and variations are possible without departing from the scope of the invention defined in the appended claims.

Claims

1. A data gathering system for use in a checkout counter to determine information relating to products to be purchased and to provide such information to a cash register system, comprising:
scale means supported within said checkout counter for determining weights of products presented to said data gathering system, said scale means including a subplatter located below the upper surface of said checkout counter,
optical scanning means supported upon said subplatter for reading coded labels on said products, and
a common interface circuit, responsive to both said scale means and said optical scanning means, for

providing weight data and coded label data to said cash register system.

2. A data gathering system for use in a checkout counter as claimed in claim 1 further comprising support means for suspending said data gathering system within said checkout counter, said scale means being secured to said support means.

5 3. A data gathering system for use in a checkout counter to determine information relating to products to be purchased, including weight data and coded label data, and to supply said information to a cash register system, said counter defining an upper surface upon which products are placed for access to said data gathering system, comprising:

support means for suspending said data gathering system within said checkout counter,

10 scale means secured to said support means for determining weights of products presented to said data gathering system, said scale means including a subplatter located below the upper surface of said checkout counter,

optical scanning means supported upon said subplatter for reading coded labels on said products, said

optical scanning means having an upper surface including an optical scanning window, said optical

15 scanning means being sized such that its upper surface is substantially aligned with the upper surface of said checkout counter when supported upon said subplatter, and

a common interface circuit, responsive to both said scale means and said optical scanning means, for providing both weight data and coded label data to said cash register system.

4. A data gathering system for use in a checkout counter as claimed in claim 1,2 or 3, wherein said
20 subplatter includes scanner locator means for positioning said optical scanning means on said subplatter for assembly of said data gathering system.

5. A data gathering system for use in a checkout counter as claimed in claim 1,2,3 or 4 in which said optical scanning means includes a bar code decoder circuit for decoding scan signals to provide coded label data, a scanner microprocessor for correlating coded label data and supplying said coded label data
25 to said common interface circuit, and memory means for storing control software for use by said scanner microprocessor.

6. A data gathering system for use in a checkout counter as claimed in claim 5, in which said scale means supplies weight data to said bar code decoder circuit of said optical scanning means, and said bar code decoder circuit supplies said weight data to said common interface circuit via said scanner
30 microprocessor without alteration.

7. A data gathering system for use in a checkout counter as claimed in any preceding claim, in which said common interface circuit comprises an interface microprocessor, responsive to coded label data from said optical scanning means and to weight data from said scale means, memory means for storing control software for use by said interface microprocessor, and a driver circuit, responsive to said interface
35 microprocessor, for supplying weight data and coded label data to said cash register system.

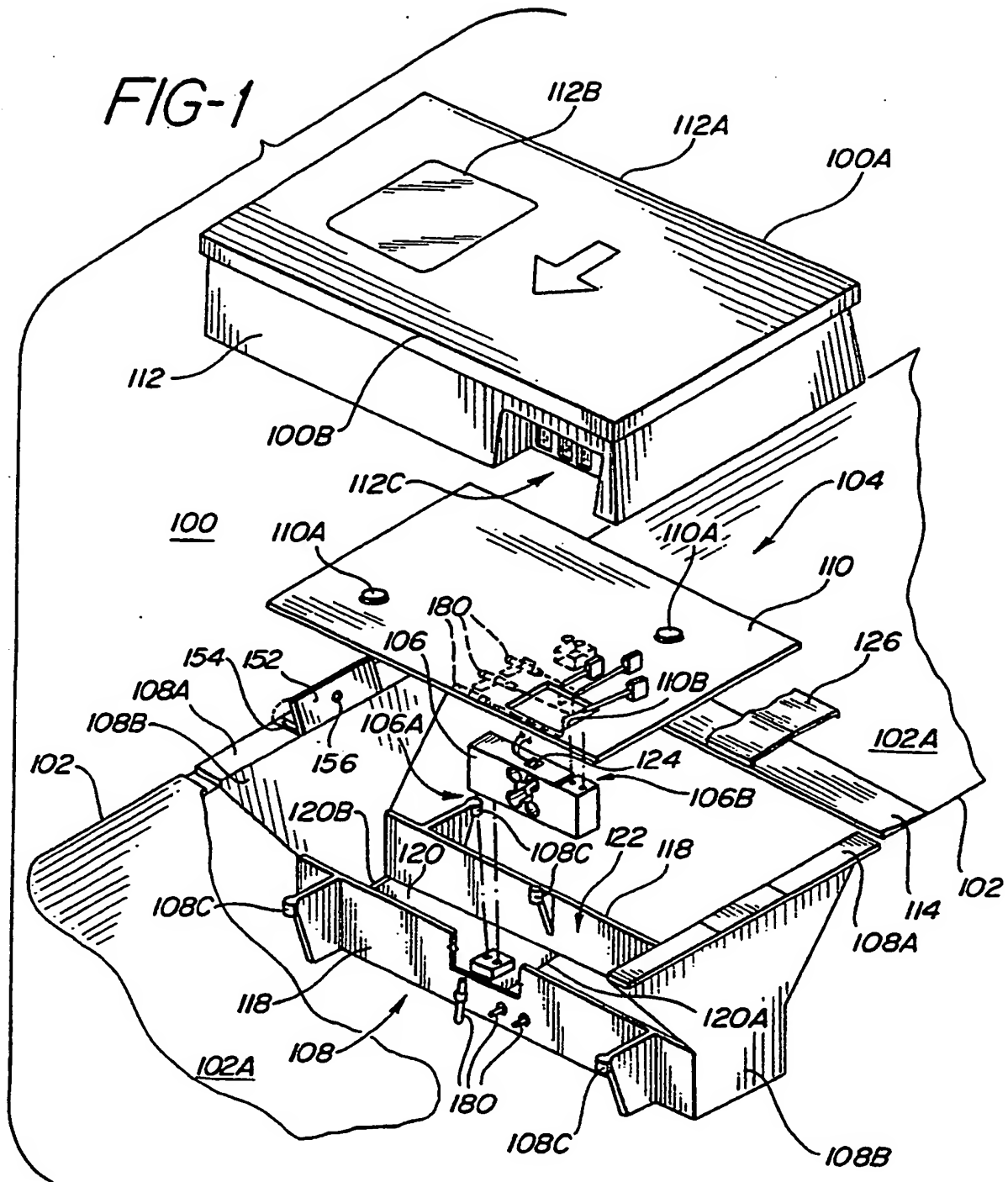
8. A data gathering system for use in a checkout counter as claimed in claim 6 or 7 in which said scale means supplies weight data directly to said common interface circuit.

9. A data gathering system for use in a checkout counter as claimed in claim 6,7 or 8, further comprising cables connected between said scale means and said optical scanning means for conducting
40 electrical signals and power, said cables being sized, positioned and secured to prevent interference with the operation of said scale means.

45

50

55



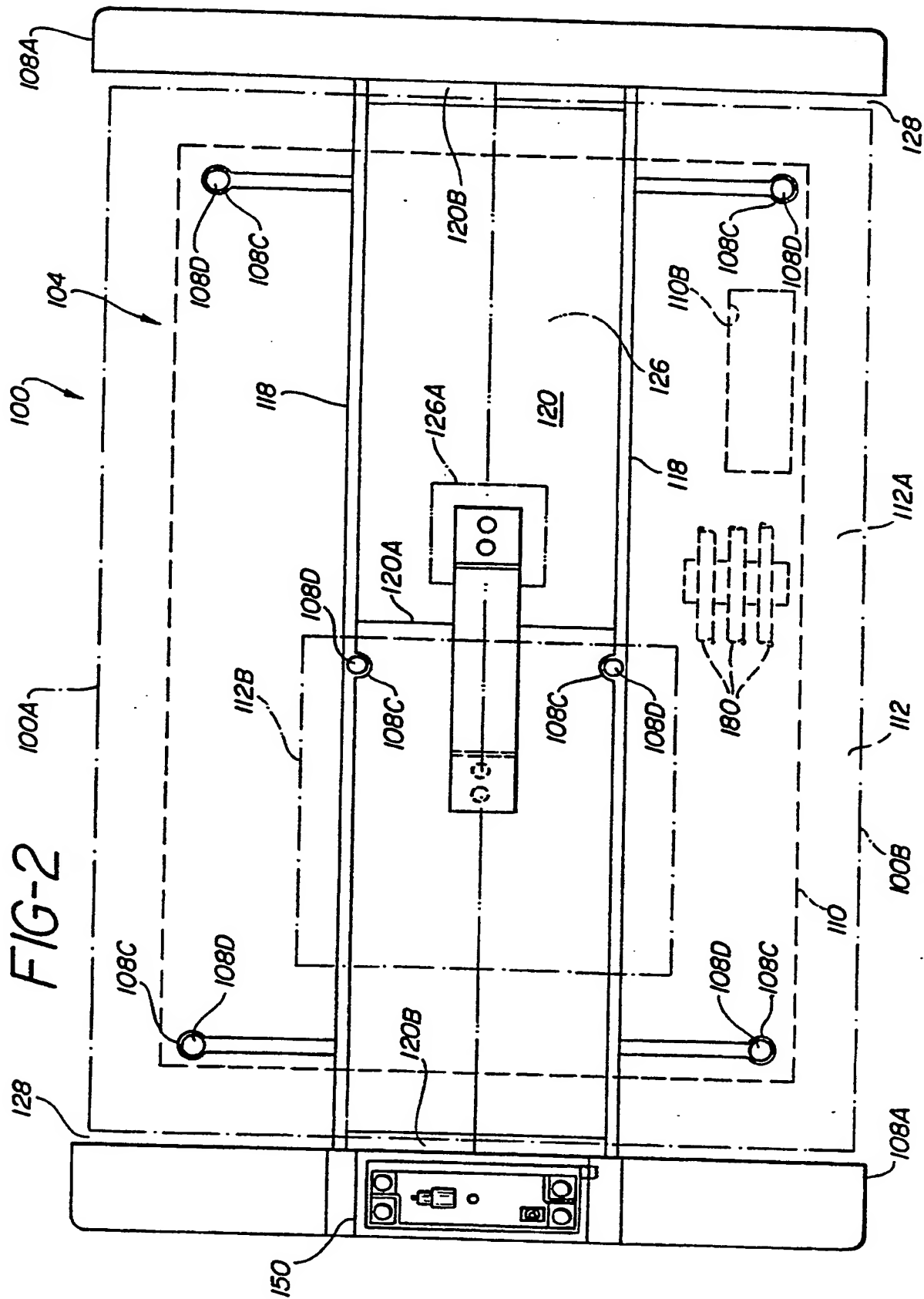
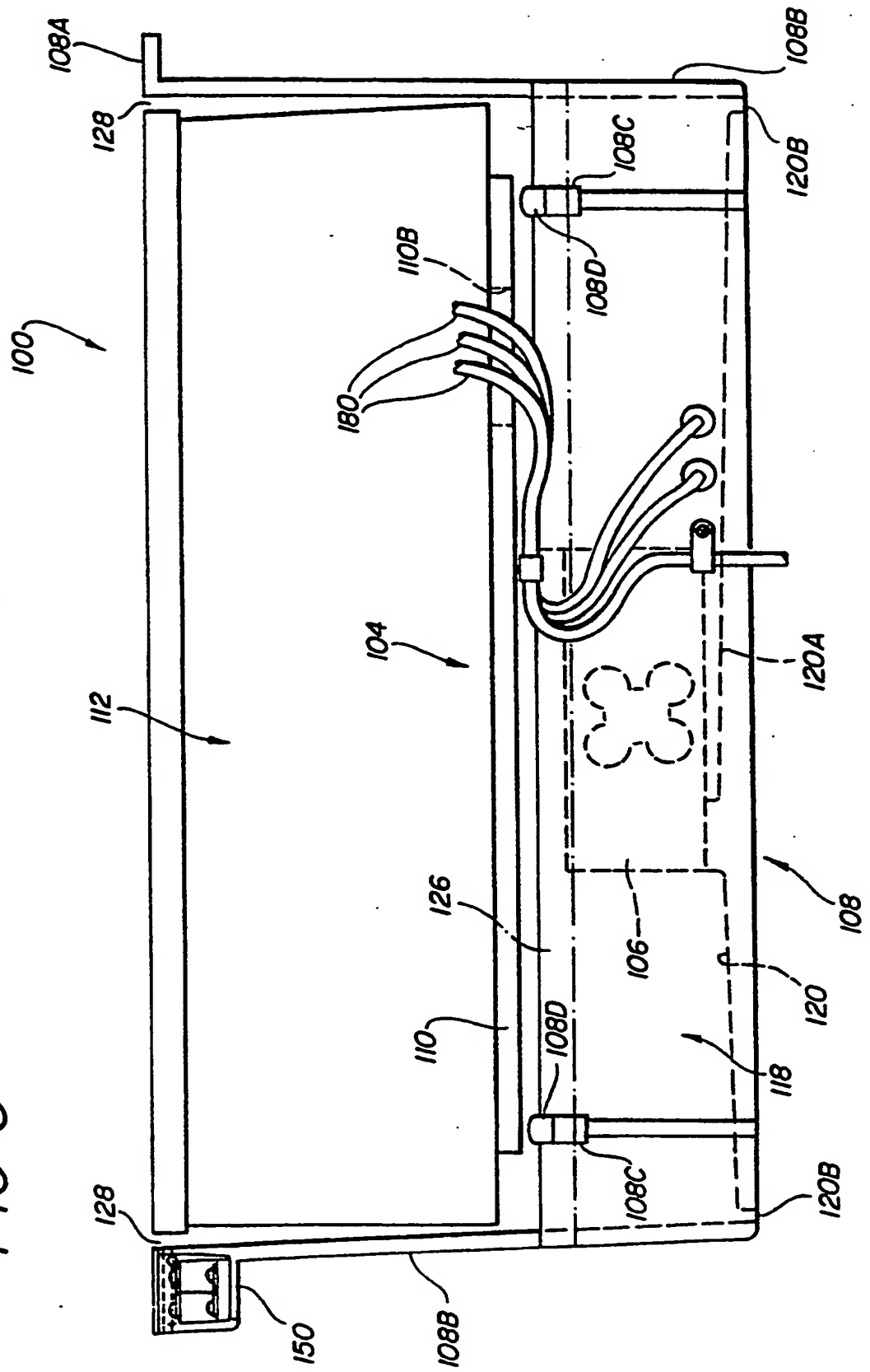


FIG-3



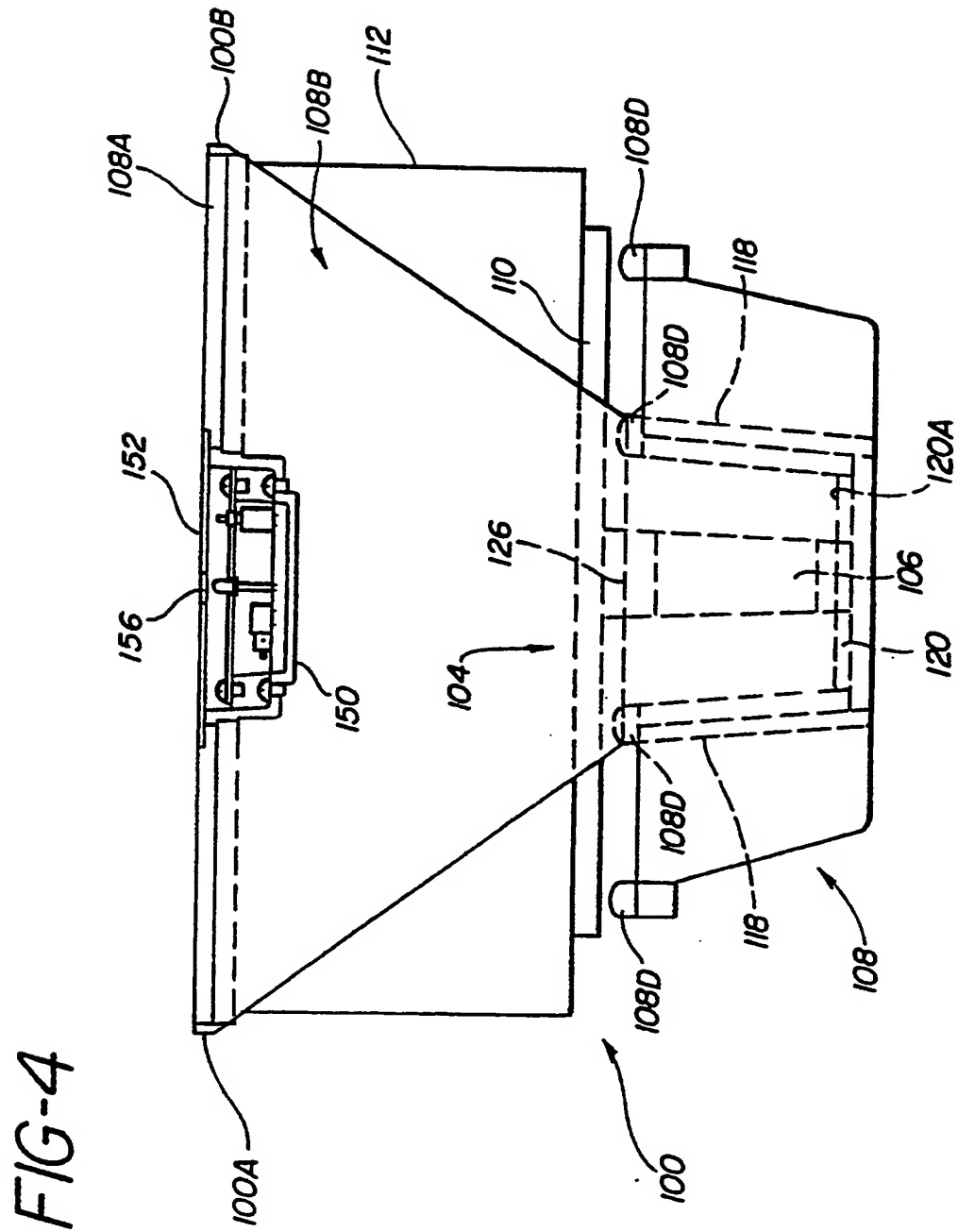
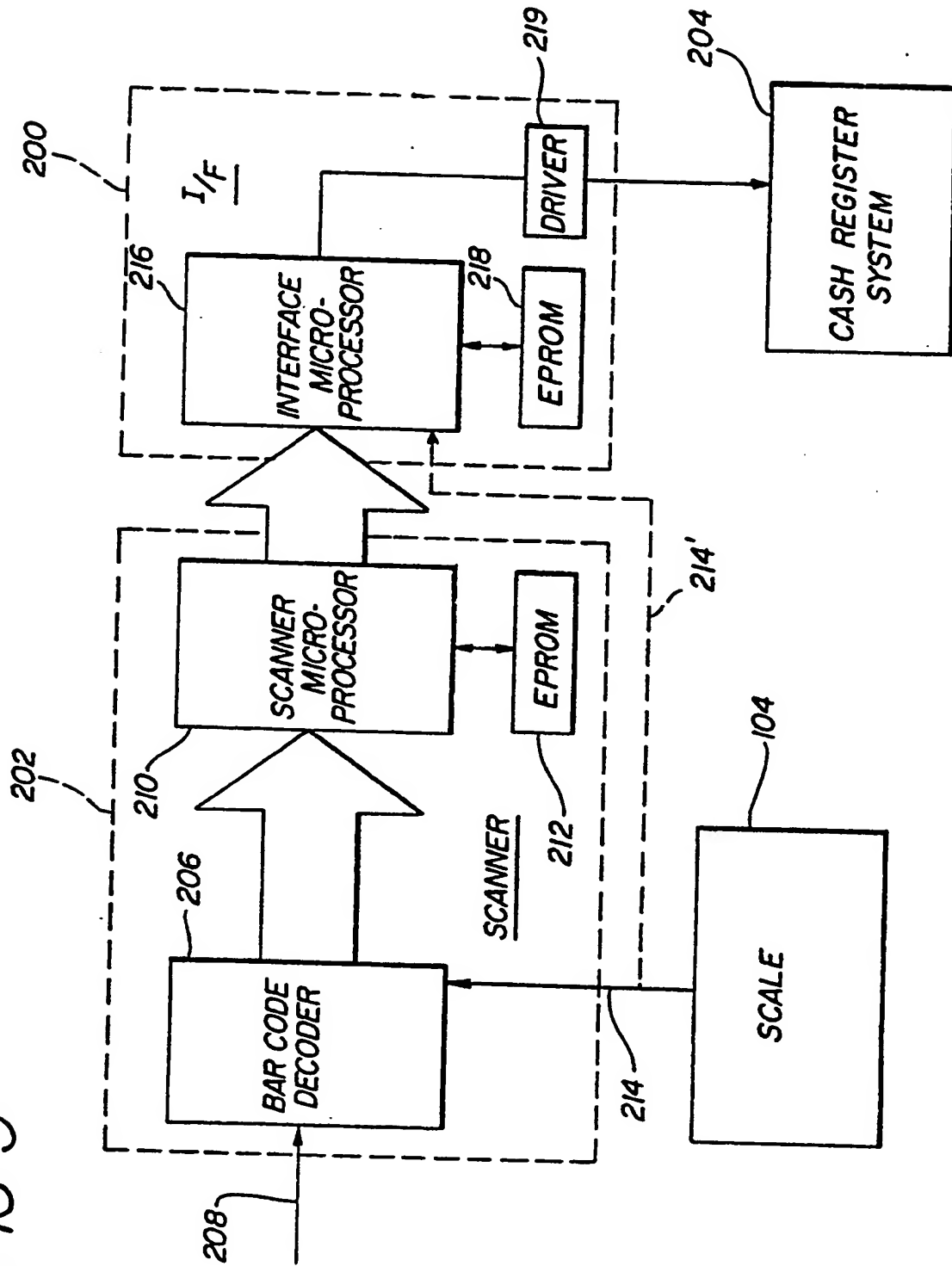


FIG-5



(19)



Europäisches Patentamt
European Patent Office
Office européen des brevets



(11) Publication number:

0 388 560 A3

(12)

EUROPEAN PATENT APPLICATION

(21) Application number: 89313390.0

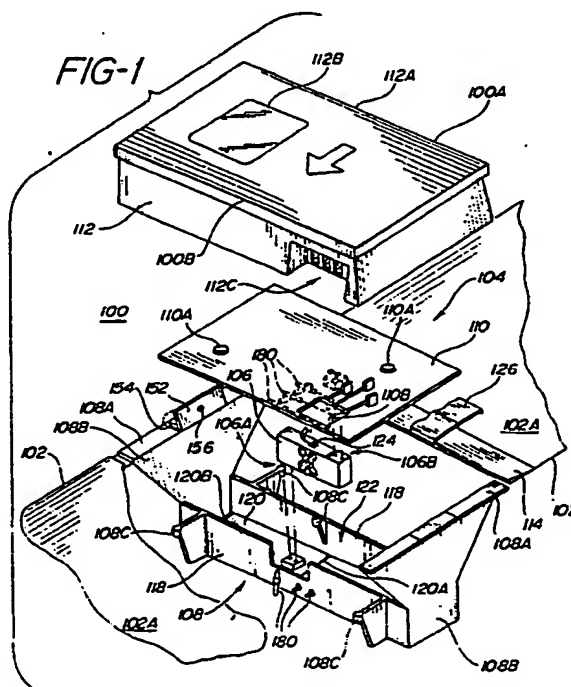
(51) Int. Cl.⁵: **G07G 1/12, G06K 7/10,
G01G 19/413**

(22) Date of filing: 20.12.89

(30) Priority: 24.03.89 US 328177

(43) Date of publication of application:
26.09.90 Bulletin 90/39(64) Designated Contracting States:
BE CH DE FR GB IT LI LU NL SE(68) Date of deferred publication of the search report:
02.01.91 Bulletin 91/01(71) Applicant: **SPECTRA PHYSICS INC.**
3333 North First Street
San Jose California 95134-1995(US)(72) Inventor: **Taussig, Andrew Peter**
2515 West 22nd Avenue
Eugene Oregon 97405(US)
Inventor: **Isaacs, Blake L.**
748 Granite Place
Springfield Oregon 97477(US)(74) Representative: **Murgatroyd, Susan Elizabeth**
et al
Baron & Warren 18 South End
Kensington
London W8 5BU(GB)(54) **Data gathering system interface.**

(57) A data gathering system (100) for use in a checkout counter (102) to determine information relating to products to be purchased and to provide such information to a cash register system includes a scale (104) supported within the checkout counter (102) for determining weights of products presented to the data gathering system (100). The scale (104) includes a subplatter (110) located below the upper surface of the checkout counter (102). An optical scanning arrangement (112) is supported on the subplatter (110) for reading coded labels on the products. A common interface circuit (200) is responsive to both the scale (104) and the optical scanning arrangement (112) for providing weight data and coded label data to the cash register system.

**EP 0 388 560 A3**



European
Patent Office

EUROPEAN SEARCH REPORT

Application Number

EP 89 31 3390

DOCUMENTS CONSIDERED TO BE RELEVANT			
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int. Cl.5)
X,X,P	JP-A-6 210 839 (TOKYO ELECTRIC CO.) * the whole document & US-A-4879650 (KURIMOTO ET.AL.) 07 November 1989 * -----	1-9	G 07 G 1/12 G 06 K 7/10 G 01 G 19/413 G 01 G 19/415
X,A	US-A-4 716 281 (AMACHER ET.AL.) * abstract; claims 1-9; figures 1-8 ** column 1, line 28 - column 7, line 9 * -----	1,2-9	
D,A	WO-A-8 605 270 (NCR CORPORATION) * the whole document * -----	1-9	
A	WO-A-8 804 813 (NCR CORPORATION) * abstract; claim 4; figures 1-3 * -----	1-6	
A	EP-A-0 168 627 (TOKYO ELECTRIC CO.) * abstract; claims 1-5; figures 1-5 * -----	1-5	
A	EP-A-0 178 223 (GRO-EST) * abstract; claims 1-3; figures 1, 2 * -----	1	
			TECHNICAL FIELDS SEARCHED (Int. Cl.5)
			G 07 G G 06 K G 01 G A 47 F
The present search report has been drawn up for all claims			
Place of search The Hague		Date of completion of search 08 November 90	Examiner GUIVOL,O.
<div>CATEGORY OF CITED DOCUMENTS</div> <div>X: particularly relevant if taken alone Y: particularly relevant if combined with another document of the same category A: technological background O: non-written disclosure P: intermediate document T: theory or principle underlying the invention</div> <div>E: earlier patent document, but published on, or after the filing date D: document cited in the application L: document cited for other reasons ----- &: member of the same patent family, corresponding document</div>			

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

THIS PAGE BLANK (USPTO),